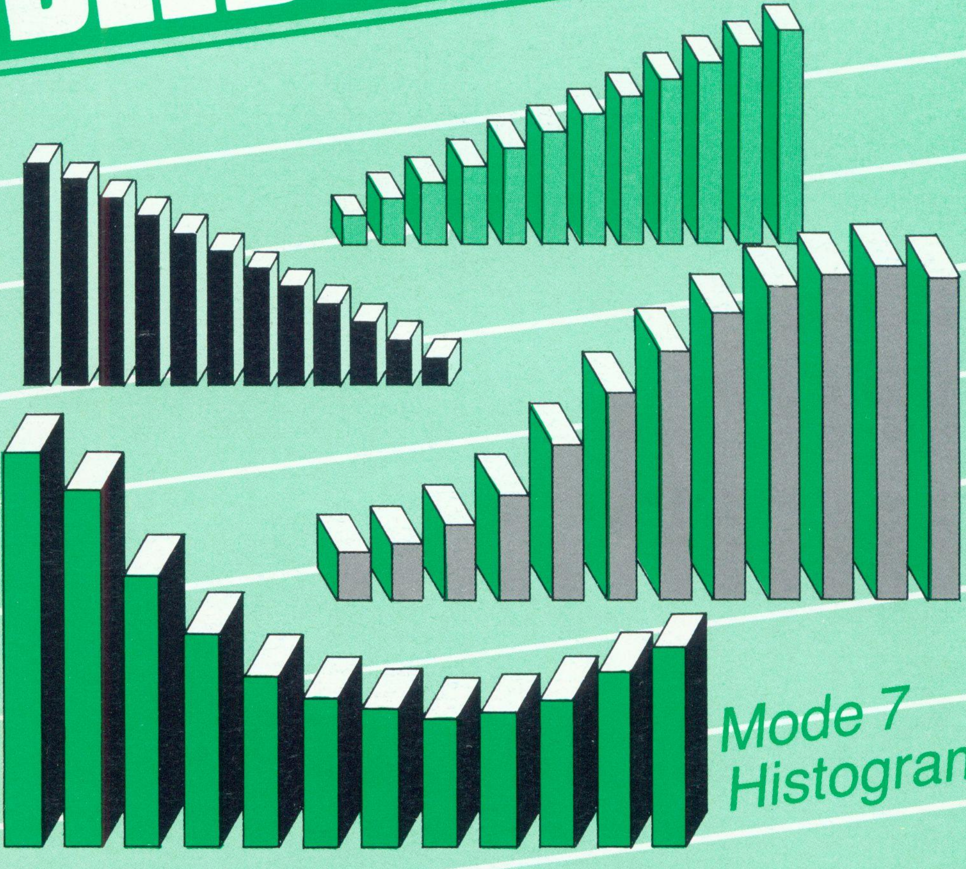
Vol.7 No.10 April 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Mode 7 Histograms

- LASER PRINTERS
- DODECA GAME
- DFS DISC INDEXER
- AC CIRCUITS

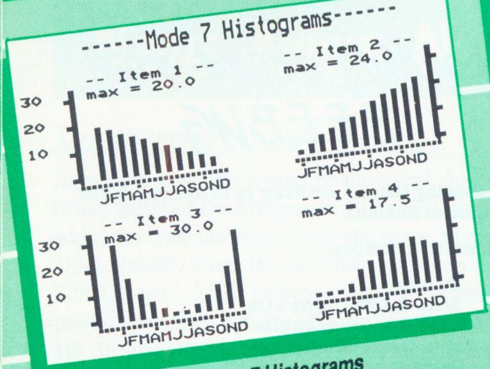
BEEBUG

Vol.7 No.10 April 1989

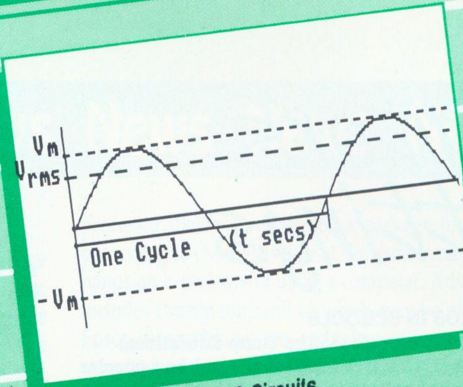
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

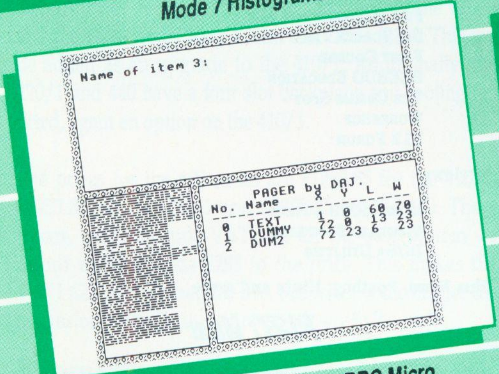
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



Mode 7 Histograms

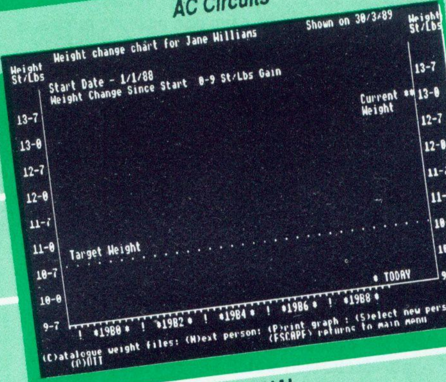


AC Circuits



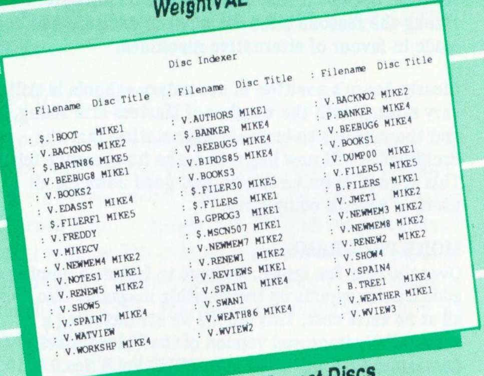
```
Name of item 3:

        PAGER by DAJ.
No. Name       X   Y   L   W
0   TEXT       1   0   60  70
1   DUMMY     72   0   13  23
2   DUM2      72  23    6  23
```

Page Composition for the BBC Micro



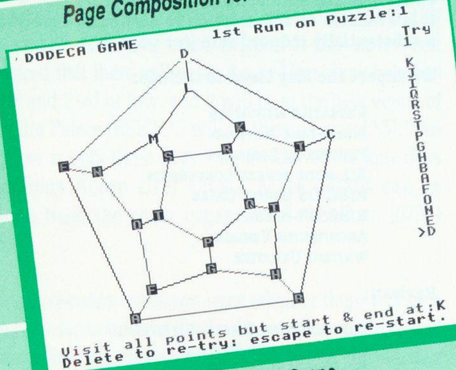
```
Weight change chart for Jane Williams          Shown on 30/3/89
Start Date - 1/1/88
Weight Change Since Start  0-9 St/Lbs Gain
(C)atalogue weight files: (N)ext person: (P)rint graph : (S)elect new person
    (Q)UIT                            (ESCAPE) returns to main menu
```

WeightVAL



```
                             Disc Indexer
: Filename  Disc Title  : Filename  Disc Title  : Filename  Disc Title
                                                : V.BACKNO2 MIKE2
                        : V.AUTHORS MIKE1       : P.BANKER  MIKE4
: $.!BOOT   MIKE1       : $.BANKER  MIKE4       : V.BEEBUG6 MIKE4
: V.BACKNOS MIKE2       : V.BEEBUG5 MIKE4       : V.BOOKS1
: $.BARTN86 MIKE5       : V.BIRDS85 MIKE4       : V.DUMP00  MIKE1
: V.BEEBUG8 MIKE1       : V.BOOKS3              : V.FILER51 MIKE5
: V.BOOKS2              : $.FILER30 MIKE5       : B.FILERS  MIKE1
: V.EDASST  MIKE4       : $.FILERS  MIKE1       : V.JMET1   MIKE2
: $.FILERF1 MIKE5       : B.GPROG3  MIKE1       : V.NEWMEM3 MIKE2
: V.FREDDY              : $.MSCN507 MIKE1       : V.NEWMEM8 MIKE2
: V.MIKECV              : V.NEWMEM7 MIKE2       : V.RENEW2  MIKE2
: V.NEWMEM4 MIKE2       : V.RENEW1  MIKE2       : V.SHOW4
: V.NOTES1  MIKE1       : V.REVIEWS MIKE1       : V.SPAIN4
: V.RENEW5  MIKE2       : V.SPAIN1  MIKE4       : B.TREE1   MIKE4
: V.SHOW5               : V.SWAN1               : V.WEATHER MIKE1
: V.SPAIN7  MIKE4       : V.WEATH86 MIKE4       : V.WVIEW3
: V.WATVIEW             : V.WVIEW2
: V.WORKSHP MIKE4
```

Indexing DFS Format Discs


DODECA GAME
1st Run on Puzzle:1
Try 1
K J I Q R S T P G H B A F O N E >D
D L K C T R S M N E O Q P G H F B A

```
Visit all points but start & end at:K
Delete to re-try: escape to re-start.
```

Dodeca Game

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BBC MICROS IN SCHOOLS
A recent survey reported in the Times Educational Supplement (17th March 1989) shows just how popular the BBC micro has been in schools. 91% of all secondary schools are said to have one or more model Bs. However, as the number of micros in schools has grown, the influence of the model B has declined, but still accounts for 30% of all micros installed. Numbers of Master 128s have been increasing rapidly, and these now account for a further 27% of micros in schools, but the Compact barely features at all.

The Archimedes is gaining ground, and has now achieved 5% of the secondary schools market, and nearly a third of all schools now have an Archimedes (one even has 30). What is quite striking, is that the Archimedes is by far the machine that schools would have preferred but didn't get. More than twice as many schools rated the Archimedes in this way compared with the Apple Macintosh, which was the next most coveted. Lack of funds, but also lack of software, were among the reasons cited for why the decision had been made in favour of alternative machines.

Clearly Acorn's position in secondary schools is still very strong, with the number of Masters still rising, and there seems to be every expectation that the Archimedes will now begin to make its presence felt. This is good news for Acorn, and good news for all users of Acorn's computers.

MORE PROGRAMS
Over the next few issues, we hope to feature a number of additional programs on the monthly magazine disc/tape, all at no extra cost. This month we are including a substantially improved version of the Video Cassette Cataloguer first published in BEEBUG Vol.6 Nos.9 & 10. The new program will handle up to a 1000 titles. Our Best of Beebug discs have also proved very popular - order yours now while supplies last (see page 51).

INDEX TO VOLUME 7
A complete reference index to all the issues of volume 7 will be sent out to BEEBUG members with the next magazine. This will mark the start of the eighth year of publication, and the eighth year of support by BEEBUG for BBC micros and their users.

This month's telesoftware password is *Hebrides*.

Looking ahead with BEEBUG

The following features are likely to be included in the May issue of BEEBUG:

Applications & Utilities:

- JULIA SETS
- SPEED & SOUND CONTROLLER
- ABBREVIATION SUBSTITUTION
- TRANSPARENT SIDEWAYS RAM LOADER
- SHARE ANALYSIS
- FILE HANDLING FOR ALL
- BIG SCREEN TEXT
- FIRST COURSE
- BEEBUG EDUCATION
- THE COMMS SPOT
- WORKSHOP
- 512 FORUM

Reviews:

- MUSIC 5000 JUNIOR
- VIEWDATA SOFTWARE
- DOS+ UTILITIES

Plus News, Postbag, Hints and more.

RISC User

**RISC User is the largest circulation magazine devoted entirely to the Acorn Archimedes range.
It is available on subscription to all BEEBUG members at a substantially reduced rate (see page 67).**

We expect the May issue to include:

- VERSATILE RENUMBER
- MULTI-TASK NOTEPAD
- PROCEDURE LIBRARIES
- ALL MODE SCREEN CONVERSION
- RISC OS SPRITE CALLS
- RISC OS BASIC
- ARCHIMEDES VISUALS
- WRITING UTILITIES

Reviews:

- PREMIER FROM CIRCLE SOFTWARE
- CLARES' RENDER BENDER
- DESIGNER INTRO FROM TECHSOFT

and more.

RISC User is the ideal magazine to keep up to date with the Archimedes scene in every respect, and is particularly useful if you are contemplating the purchase of an Archimedes in the near future.

The latest details of the contents and distribution of both magazines are contained in the BEEBUG area of Micronet. Just type *BEEBUG# when on-line.

News News News News News News

ARCHIMEDES RANGE EXPANDS

Almost two years after it was first announced, Acorn have finally launched the Archimedes 410. In fact, they have added two new machines to the 400 series, to give the 410/1, the 420/1, and the 440. These computers include RISC OS, and 1, 2, and 4 Mbyte of RAM respectively. It is possible to upgrade both the 410/1 and the 420/1 to a full 440. All three systems have the hard disc interface circuitry already fitted to the main board, though only the 420/1 and the 440 actually have a hard disc drive fitted. This can be added as an upgrade to the 410/1. Additionally, the 420/1 and 440 have a four slot backplane and cooling fan fitted, again an option on the 410/1.

The prices for the 400 series are £1263.85 for the 410/1, £1953.85 for the 420/1 and £2908.35 for the 440. These prices, which include VAT, are for the base system - a colour monitor adds £253 to the price. This makes the 410/1 £230 dearer than the 310, but when you consider the expansion potential this extra is well worth it.

SHOW MYSTERY SOLVED

Our speculation of Beeb exhibitions given in last month's news appears to have been correct. It has now been announced that there will be an Acorn User show between the 21st and 23rd of July, and it will be at the new venue of Alexandra Palace (BEEBUG will be there on stand 55). This new show is only three days long instead of the four days of previous Acorn User shows. More details can be obtained from the show organisers SafeSell on (0737) 814060.

There still remains confusion over whether there will be a Micro User show or not. Clearly the usual one in May will not take place, but there are conflicting reports of whether there will be one in November or not. Whatever happens, BEEBUG will keep you informed.

EASY WORD PROCESSING

Tedimen Software, the company that produces the Folio word processor system has upgraded the software to Advanced Folio. Rather than being a practical word processor, Folio is very much a package to teach word processing to children, while at the same time giving them hands on experience of using a computer. Advanced Folio includes twenty fonts, all of which may be printed on an Epson compatible printer, and a font editor to design your own fonts. Advanced Folio costs £33.07 (inc. VAT) and is available from most Acorn dealers, or direct from Tedimen Software, PO Box 23, Southampton, Hampshire.

BANANA BOARDS

The Banana interface from Castle Associates is an interface board for the model B and Master 128. It is primarily aimed at education, particularly the new CDT syllabuses, and allows equipment to be interfaced to the computer's user port without fear of damaging the computer. The unit is supplied complete with a connecting cable for the user port, and a comprehensive manual which includes several suggested projects which incorporate the device. The Banana interface costs £231.44 (inc. VAT), and more details can be obtained from Castle Associates, Salter Road, Scarborough, North Yorkshire YO11 3UZ, tel. (0723) 584250.

OPTICAL FACTS

For pupils revising hard for the Physics GCSE exam, help comes in the form of two resource packs from Chalksoft. The software, called Optics A and Optics C (what happened to B?) is designed to illustrate the principles of optical systems. The software costs £23 for both packs, or £25.30 for a 3.5" disc version. A pupil's workbook is available for £3.95. The packs are available from most Acorn dealers.

PRINTER POWER

The KX-P1180 is a new nine-pin dot matrix printer from Panasonic, priced at £250 (inc. VAT). The printer is fully Epson compatible, and supports IBM graphics characters. There are two draft printing modes, and a near letter quality mode which can operate in three different fonts (Courier, Prestige and Sans Serif). More details from Panasonic Industrial UK, 280 - 290 Bath Road, Slough, Berkshire SL1 6JQ, tel. (0753) 73181.

Indexing DFS Format Discs

Laurence Cox presents a short utility that will produce a printed index to the contents of all your DFS format discs.

One of the advantages of disc over tape is the ease of making backups of files; equally it becomes practical to save programs regularly during development, for example by using the *AUTOSAVE command in BEEBUG's Toolkit Plus ROM. The result of this, however, can be a jungle of files with similar names, requiring regular pruning to keep it under control and to ensure that a desired file can be found quickly.

This is not too difficult for a few discs; the disc catalogue can be printed out and stuck on to the disc envelope (David Lee's tip in Vol.6 No.8 gives a function key string to do this). More useful for libraries of several tens of discs is a sorted list of all the files on every disc; in other words a disc index.

USING THE PROGRAM

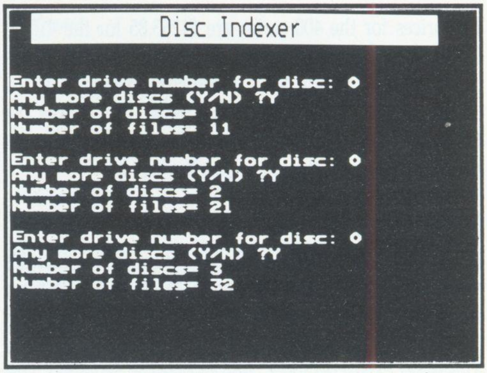
The disc indexing program listed here runs in mode 7 to give the maximum memory space for filenames. Those who have a Master, B+ or a shadow RAM board fitted can, of course, run the program in an 80-column screen mode and still increase the size of the string array used for the filenames. Make sure you have a copy of the program saved before you start to use it.

The program will ask you for the drive number for each disc which you want to index, and will then read all the information it requires directly from the disc itself. Once all your discs have been read, the program proceeds to sort and print the disc index.

UNDERSTANDING THE PROGRAM

Files with the same or similar names can arise from several causes: one disc may be copied from another using *BACKUP; individual files or groups of files may be copied with *COPY; either the final character of a filename or the directory can be incremented to denote a later version of a file.

The program sorts the files alphabetically by filename. Files with the same filename but different directories are sorted alphabetically by directory. Files with the same filename and directory but on different discs are sorted in the order that the discs were read by the program. Although it does not affect the running of the program, the user should ensure that all disc surfaces have unique titles. Use the *TITLE command to do this before cataloguing the discs. Otherwise the disc title will appear blank in the printout, which will be confusing to say the least.



```
Disc Indexer

Enter drive number for disc: 0
Any more discs (Y/N) ?Y
Number of discs= 1
Number of files= 11

Enter drive number for disc: 0
Any more discs (Y/N) ?Y
Number of discs= 2
Number of files= 21

Enter drive number for disc: 0
Any more discs (Y/N) ?Y
Number of discs= 3
Number of files= 32
```

Program in operation

The program operates by cataloguing the disc surface selected by the user, storing the disc title in the array TITLE$, and then appending the character "#" and the index number of the disc title to each filename, and storing these in the array FILE$. The program repeats the PROCreaddisc procedure until the number of disc surfaces reaches 50, or the number of files on the disc to be catalogued would cause the total to exceed 1000, or the user terminates the routine.

To read the disc catalogue, the program uses an OSWORD call to read the first two 256-byte sectors on the disc into memory. These contain the disc catalogue from which the program extracts all the information on file names and directories which it needs.

Next, the filenames are sorted using a Shell sort, with the zeroth element of FILE$ being used to exchange the pairs of strings. Lastly, the sorted filenames are printed out in three columns with their corresponding disc titles. The page layout was set up for an Epson FX80 using 11-inch fanfold paper, but should be satisfactory for virtually all printers.

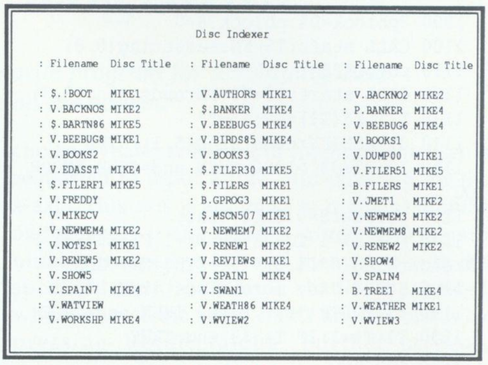
MODIFYING THE PROGRAM

The program has been structured to make it easy to modify; the variables are listed below. Those variables used for loop control and address calculation are defined as LOCAL in each procedure to clarify their scope, but these statements can be removed without affecting the operation of the program if more space is needed.

Multi-statement lines have been avoided, except where the statements form a logical unit. Further space may be saved at the cost of readability by removing all REMs, lines with only a colon and by making more multi-statement lines but some care is necessary to avoid the "No room" message when subsequently increasing the string array size.

VARIABLES LIST

A%, B%, C%, I%, J%, K%, L%, M%	Local variables used for loop control and calculating locations in memory.
D%	Drive number of disc to be read
F%	Number of files
T%	Number of disc titles
line%	Number of lines of filenames on page
col%	Number of columns of filenames on page
T$,F$	Used to build up titles and filenames character by character
TITLE$(49)	Array of disc titles
FILE$(1000)	Array of disc filenames
p$	String in which filenames and disc titles are built up.
Shead$	Title for screen display.
Phead$	Header for printout.
noswap	
end	Logical loop control variables



```
                         Disc Indexer

 : Filename  Disc Title   : Filename  Disc Title   : Filename  Disc Title

 : $.!BOOT   MIKE1        : V.AUTHORS MIKE1        : V.BACKNO2 MIKE2
 : V.BACKNOS MIKE2        : $.BANKER  MIKE4        : P.BANKER  MIKE4
 : $.BARTN86 MIKE5        : V.BEEBUG5 MIKE4        : V.BEEBUG6 MIKE4
 : V.BEEBUG8 MIKE1        : V.BIRDS85 MIKE4        : V.BOOKS1
 : V.BOOKS2               : V.BOOKS3               : V.DUMP00  MIKE1
 : V.EDASST  MIKE4        : $.FILER30 MIKE5        : V.FILER51 MIKE5
 : $.FILERF1 MIKE5        : $.FILERS  MIKE1        : B.FILERS  MIKE1
 : V.FREDDY               : B.GPROG3  MIKE1        : V.JMET1   MIKE2
 : V.MIKECV               : $.MSCN507 MIKE1        : V.NEWMEM3 MIKE2
 : V.NEWMEM4 MIKE2        : V.NEWMEM7 MIKE2        : V.NEWMEM8 MIKE2
 : V.NOTES1  MIKE1        : V.RENEW1  MIKE2        : V.RENEW2  MIKE2
 : V.RENEW5  MIKE2        : V.REVIEWS MIKE1        : V.SHOW4
 : V.SHOW5                : V.SPAIN1  MIKE4        : V.SPAIN4
 : V.SPAIN7  MIKE4        : V.SWAN1                : B.TREE1   MIKE4
 : V.WATVIEW              : V.WEATH86 MIKE4        : V.WEATHER MIKE1
 : V.WORKSHP MIKE4        : V.WVIEW2               : V.WVIEW3
```

Printed listing

```
   10 REM Disc Indexing Program
   20 REM Version B1.1
   30 REM Author  Laurence Cox
   40 REM BEEBUG  April 1989
   50 REM Program subject to copyright
   60 :
  100 MODE7:ON ERROR GOTO 240
  110 PROCinit
  120 PROCtitle
  130 PROCassemble
  140 REPEAT
  150 PROCreaddisc
  160 PRINT "Number of discs= ";T%
  170 PRINT "Number of files= ";F%
  180 UNTIL end
  190 PROCsortfiles
  200 PROCprintfiles
  210 p=VPOS:VDU26,31,0,p+3
  220 END
  230 :
  240 MODE7:VDU3:*FX3,0
  250 REPORT:PRINT" at line ";ERL
  260 END
  270 :
 1000 DEF PROCtitle
 1010 PRINTTAB(0,0)CHR$141CHR$130;CHR$15
7;CHR$132;TAB(13);Shead$;TAB(37)CHR$156
 1020 PRINTTAB(0,1)CHR$141CHR$130;CHR$15
7;CHR$132;TAB(13);Shead$;TAB(37)CHR$156
 1030 VDU28,0,24,39,3
 1040 ENDPROC
 1050 :
 1060 DEF PROCreaddisc
 1070 LOCAL I%,J%,K%,L%
 1080 INPUT'"Enter drive number for disc
: "D%
```

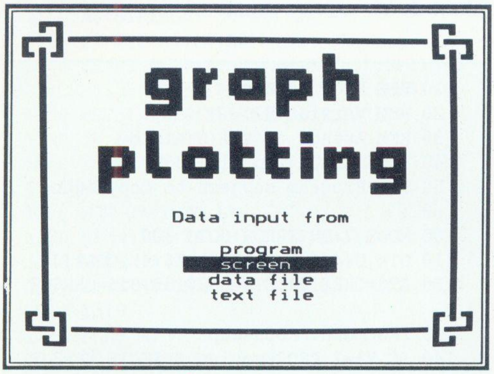
```
 1090 ?pblock=D%:pblock?8=0
 1100 CALL start:T$=FNreadstring(0,8)
 1110 pblock?8=1
 1120 CALL start:T$=T$+FNreadstring(0,4)
 1130 TITLE$(T%)=T$:T$=""
 1140 J%=ASC(FNreadstring(5,1))DIV8
 1150 IF F%+J% >1000 THEN end=TRUE:ENDPR
OC
 1160 pblock?8=0:CALL start
 1170 FOR I%=1 TO J%:F%=F%+1:FILE$(F%)=F
Nreadstring(8*I%,8)+"#"+STR$(T%):NEXT
 1180 INPUT "Any more discs (Y/N) ",Y$
 1190 IF INSTR("Yy",Y$)=0 THEN end=TRUE
 1200 T%=T%+1:IF T%>49 end=TRUE
 1210 ENDPROC
 1220 :
 1230 DEF PROCsortfiles
 1240 LOCAL A%,B%,C%
 1250 A%=1023:REPEAT:A%=INT(A%/2):UNTIL
A%<F%
 1260 REPEAT
 1270 REPEAT:noswap=TRUE
 1280 FOR B%=1 TO F%-A%
 1290 C%=B%+A%
 1300 IF FILE$(C%) < FILE$(B%) THEN FILE
$(0)=FILE$(C%):FILE$(C%)=FILE$(B%):FILE$
(B%)=FILE$(0):noswap=FALSE
 1310 NEXT
 1320 UNTIL noswap
 1330 A%=INT(A%/2)
 1340 UNTIL A%=0
 1350 ENDPROC
 1360 :
 1370 DEF PROCprintfiles
 1380 LOCAL I%,J%,K%,L%,M%
 1390 PRINT'"Printing index"
 1400 *FX3,10
 1410 VDU2:PRINTSPC(30) Phead$'
 1420 FOR K%=1 TO F%
 1430 L%=INSTR(FILE$(K%),"#")
 1440 IF MID$(FILE$(K%),L%-1,1)=CHR$(32)
 THEN p$=p$+": $." ELSE p$=p$+": "+MID$(
FILE$(K%),L%-1,1)+"."
 1450 p$=p$+LEFT$(FILE$(K%),L%-2)
 1460 M%=VAL(MID$(FILE$(K%),L%+1))
 1470 p$=p$+" "+TITLE$(M%)+" "
 1480 IF (K% MOD col%)=0 THEN PRINT p$:p
$="    "
 1490 IF (K% MOD (line%*col%))=0 PRINT '
SPC(30)"Page ";(K% DIV (line%*col%)):VDU
12:PRINT SPC(30)Phead$'
 1500 NEXT
 1510 PRINT p$
 1520 J%=55-(((K% DIV col%) MOD line%)+1
)
 1530 FOR I%=1 TO J%:PRINT:NEXT
 1540 PRINT SPC(30)"Page ";((K% DIV (lin
e%*col%))+1):VDU12,3
 1550 *FX3,0
 1560 PRINT'"Printing complete"
 1570 ENDPROC
 1580 :
 1590 DEF PROCinit
 1600 *DISC
 1610 *DIR $
 1620 end=FALSE:T%=0:F%=0:line%=53:col%=
3
 1630 DIM TITLE$(49)
 1640 DIM FILE$(1000)
 1650 T$=STRING$(8," ")::T$="":p$=STRING
$(72," "):p$=STRING$(4," ")
 1660 Shead$="Disc Indexer"
 1670 Phead$=Shead$+CHR$(13)+CHR$(10)+CH
R$(13)+CHR$(10)+"      "+STRING$(3,": File
name  Disc Title    ")
 1680 ENDPROC
 1690 :
 1700 DEF PROCassemble
 1710 OSWORD=&FFF1
 1720 DIM names 256, code% 25
 1730 FOR pass%=0 TO 2 STEP 2
 1740 P%=code%
 1750 [OPT pass%
 1760 .pblock EQUB 0
 1770 EQUD names
 1780 EQUB &03
 1790 EQUB &53
 1800 EQUB &00
 1810 EQUB &00
 1820 EQUB &21
 1830 .res EQUB0
 1840 .pb EQUW pblock
 1850 .start LDA #&7F
 1860 LDX pb
 1870 LDY pb+1
 1880 JSR OSWORD
 1890 RTS
 1900 ]
 1910 NEXT pass%
 1920 ENDPROC
 1930 :
 1940 DEF FNreadstring(s%,n%)
 1950 LOCAL I%,p$:p$=""
 1960 FOR I%=s% TO s%+n%-1
 1970 p=?(names+I%)AND&7F
 1980 IF p>96 AND p<123 p=p AND&5F
 1990 IF p=0 p=32
 2000 p$=p$+CHR$(p)
 2010 NEXT I%
 2020 =p$
```

Mode 7 Histograms

Carol Stainsby describes a highly versatile program for displaying data in histogram form on the screen.

Mode 7 is not normally used to portray graphical data, but this program, displaying up to four items each on separate histograms, indicates how effectively this mode may be used.

The screen is divided into quadrants, each one depicting a small histogram of up to twelve data values. The y-axis of each plot has the same scale (with automatic scaling) thus allowing a visual comparison of either four items over the same x-values, or one item over four different ranges of x-values.



```
graph
plotting

Data input from

program
screen
data file
text file
```

Menu screen

USING THE PROGRAM

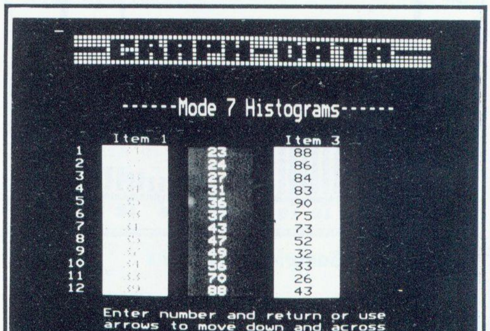
Type in the program and save to disc or tape. When run, an initial menu screen is used to select the medium from which the data values are read. This may be from:

1. the program, as DATA statements
2. the screen, by filling in a form
3. a data file - the sample procedure uses a FILER file (the BEEBUG magazine database).
4. a text file (created with a word processor such as View or Wordwise).

Use the up/down cursor keys to select a menu option, followed by Return to confirm that choice. Whatever the input medium, the user is requested to input a title, and labelling for the x-axis. This axis is assumed to apply to months of the year, if the label is given as a month number (range 1 to 12), and then starts with the specified month. Default headings are used where any of the reply fields are left blank by pressing Return alone. Further screens, requesting information, are then displayed according to the source of data chosen, before the histogram display appears.

DATA INPUT

If data is to be supplied as DATA statements then these should be placed at lines 1710 to 1740, as with the demonstration data included in the program listing. Each DATA statement should contain 12 values for one variable. If less than 12 values are required, include zeros where appropriate to maintain the correct total (or vary the loop parameters in lines 1670 and 1680).



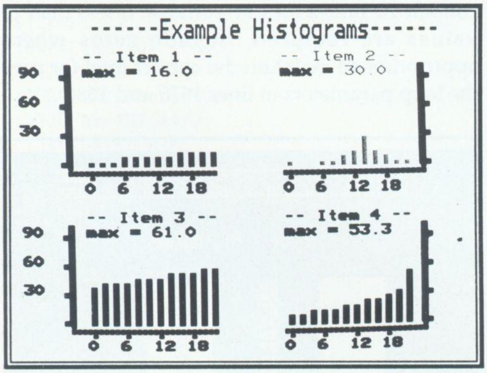
```
GRAPH-DATA

------Mode 7 Histograms------

      Item 1              Item 3
 1                  23      88
 2                  24      86
 3                  27      84
 4                  31      83
 5                  36      90
 6                  37      75
 7                  43      73
 8                  47      52
 9                  49      32
10                  56      33
11                  70      26
12                  88      43

Enter number and return or use
arrows to move down and across
```

Screen data entry

For screen input, four initially blank columns are displayed, one for each variable. Press Return or cursor down after each value, and use cursor right to move to the next column. At the end of data input, cursor right or Return will lead to the graph display. Values left blank on input will be taken as zeros.

The program will also accept data from a data file created with the Filer database program. This program was described in BEEBUG Vol.4 Nos.6-8, with additional supporting programs in later issues (Vol.5 Nos.2 & 4 for example, and most recently Vol.6 No.5). You will be asked to specify the name of your data file and drive number (if no drive number is specified, the current drive is assumed), and the names of any four fields in any order. The program will tell you the number of records in the file and ask you to specify at which record to start to read the twelve sets of data. The four graphs will then be displayed as before.

Lastly, a data file (in ASCII text format) can be created with any suitable word processor. Numbers should be entered in four lines of twelve, with numbers separated by commas or Return. Such a file can be easily edited when required.



Sample histograms

NOTE: The complete suite of Filer programs with documentation is still available for just £5.50 plus £1 p&p. This month's magazine disc/tape contains a sample data file of 20 records using the field names "ITEM1", "ITEM2", "ITEM3", "ITEM4", and "ITEM5".

PROGRAM NOTES

As it stands the program will only cope with data values in the range 0 to 9999. This is checked for values entered via the screen, but if the existing DATA statements are amended, or a Filer or text file is being used, care must be taken to ensure that numbers are kept within these limits.

Lines 2720 to 3070 are concerned solely with screen input, lines 3090 to 3430 with file input, and lines 3450 to 3730 with text file input, so the graphical output can be viewed by omitting all or any of these sections, and choosing *program* from the menu as the data source. The default headings and colours are initialised in PROCsetup and can be amended easily. Alphanumeric colour codes are given at line 1290 and the corresponding graphic colours are calculated from these values. The odd looking character strings which appear in the program represent mode 7 graphics characters.

```
   10 REM Program Hist
   20 REM Version B1.6
   30 REM Author  Carol Stainsby
   40 REM BEEBUG  April 1989
   50 REM Program subject to copyright
   60 :
  100 MODE 7:ON ERROR GOTO 300
  110 DIM D(4,12),GMAX(4),T$(4),COL%(4)
  120 CT$=CHR$134:GT$=CHR$130:BT$=CHR$13
2
  130 PROCmenu:PROCsetup
  140 IF Y%=2 PROCproginp ELSE IF Y%=3 P
ROCscrninp ELSE IF Y%=4 PROCfileinp ELSE
 PROCtextinp
  150 FOR I%=1 TO 4
  160 FOR J%=1 TO 12
  170 d=D(I%,J%)
  180 IF d>MAX THEN MAX=d
  190 IF d>GMAX(I%) THEN GMAX(I%)=d
  200 NEXT:NEXT
  210 PROCskel:PROCyscale
  220 RESTORE 270
  230 FOR G%=1 TO 4
  240 READ x%,y%:PROCgraph(G%,x%,y%)
  250 NEXT
  260 VDU 31,0,24,132,157,134
  270 DATA 8,10,26,10,8,21,26,21
  280 END
  290 :
  300 @%=10:*FX4,0
  310 MODE7:REPORT:PRINT" at line ";ERL
  320 END
  330 :
 1000 DEF PROCsetup
 1010 CLS:MAX=0
 1020 T$(1)="-- Item 1 --"
 1030 T$(2)="-- Item 2 --"
 1040 T$(3)="-- Item 3 --"
 1050 T$(4)="-- Item 4 --"
 1060 RESTORE 1290
 1070 FOR I%=1 TO 4
 1080 READ COL%(I%):GMAX(I%)=0
 1090 FOR J%=1 TO 12
 1100 D(I%,J%)=0
 1110 NEXT:NEXT
 1120 PRINT CT$,"Headings data"'GT$;"Tit
le:"'CHR$157;BT$;STRING$(28,".");CHR$156
'CT$;"  (] gives default heading)"
 1130 VDU31,2,2:t$=FNinput(28,"A")
 1140 IF L%=0 t$="Mode 7 Histograms":L%=
17
 1150 A$=STRING$((30-L%)/2,"-")
 1160 t$=CT$+A$+t$+A$
 1170 PRINT 'GT$;"       x-scale:";SPC16'S
PC9;CHR$157;BT$;STRING$(12,".");CHR$156;
 1180 VDU31,11,4:X$=FNinput(12,"A")
 1190 IF L%<3 GOTO1210
 1200 X$=X$+STRING$(12-L%," "):GOTO 1310
 1210 M%=0
 1220 FOR J%=1 TO L%
 1230 M%=M%*10+VAL(MID$(X$,J%,1))
 1240 NEXT
 1250 IF M%=0 OR M%>12 X$="0  6  12 18 "
:GOTO 1310
 1260 READ M$
 1270 R$=RIGHT$(M$,13-M%)
 1280 L$=LEFT$(M$,M%-1):X$=R$+L$
 1290 DATA 146,149,150,145
 1300 DATA JFMAMJJASOND
 1310 ENDPROC
 1320 :
 1330 DEF PROCskelv
 1340 VDU31,5,Y%,151,250,C%,31,25,Y%,c%,
31,38,Y%,151,245
 1350 Y%=Y%+1
 1360 VDU31,5,Y%,151,234,C%,31,25,Y%,c%,
31,38,Y%,151,181
 1370 Y%=Y%+1
 1380 ENDPROC
 1390 :
 1400 DEF PROCskel
 1410 VDU26,12:*FX4,0
 1420 A$="    "+CHR$132+CHR$141+CHR$157+
t$
 1430 PRINT A$'A$:A$="max ="
 1440 @%=&00020106:L%=1
 1450 FOR K%=11 TO 22 STEP 11
 1460 N%=L%+1:Y%=K%-7
 1470 C%=COL%(L%):c%=COL%(N%)
 1480 TC%=C%-16:tc%=c%-16
 1490 PRINTTAB(7,Y%-1);CHR$(TC%);T$(L%);
SPC(5);CHR$(tc%);T$(N%)
 1500 PRINTTAB(8,Y%);A$;CHR$(TC%);GMAX(L
%);
 1510 PRINTTAB(26,Y%);A$;CHR$(tc%);GMAX(
N%)
 1520 FOR I%=1 TO 4
 1530 PROCskelv
 1540 NEXT
 1550 VDU31,4,K%,151,154,170,172
 1560 PRINT STRING$(4,"l,,")
 1570 VDU31,25,K%,151
 1580 PRINT STRING$(4,"l,,");
 1590 VDU172,165
 1600 PRINT SPC(8);X$;SPC(6);X$
 1610 L%=L%+2
 1620 NEXT:@%=10
 1630 ENDPROC
 1640 :
 1650 DEF PROCproginp
 1660 RESTORE 1710
 1670 FOR I%=1 TO 4
 1680 FOR J%=1 TO 12
 1690 READ D(I%,J%)
 1700 NEXT:NEXT
 1710 DATA 19.99,18.33,16.66,15.0,13.33,
11.66,10.0,8.33,6.66,5.0,3.33,0
 1720 DATA 2,4,6,8,10,12,14,16,18,20,22,
24
 1730 DATA 30,16,10,6,3,2,2,3,6,10,16,30
 1740 DATA 0.9,1.2,2.0,4.0,8.3,11.7,15.0
,16.5,17.3,17.5,17.0,15.0
 1750 ENDPROC
 1760 :
 1770 DEF PROCyscale
 1780 I%=1:J%=10
 1790 REPEAT
 1800 V=MAX/I%:I%=I%*J%
 1810 UNTIL V<J%
 1820 I%=I%/J%:W=V/3:J%=W+0.5
 1830 IF J%*3 < V J%=J%+1
 1840 J%=J%*I%:MAX%=J%*3
 1850 @%=&00000404
 1860 FOR I%=4 TO 15 STEP 11
 1870 PRINT TAB(0,I%),MAX%;TAB(7,I%);CHR
$(135);TAB(25,I%);CHR$(135)
 1880 Y%=I%+4:PRINT TAB(0,Y%),J%
 1890 K%=J%+J%:Y%=I%+2
 1900 PRINTTAB(0,Y%),K%
 1910 NEXT
 1920 @%=10:V=MAX%/18
 1930 ENDPROC
 1940 :
 1950 DEF PROCgraph(I%,X%,Y%)
 1960 LOCAL x%
 1970 FOR J%=1 TO 12
 1980 W=D(I%,J%)
 1990 IF W=0 GOTO2070
 2000 W=W/V+0.5:W%=W
 2010 x%=X%+J%-1:VDU31,x%,Y%
 2020 IF W%<1 THEN GOTO 2070
 2030 K%=234
 2040 IF W%<3 THEN K%=232
 2050 IF W%<2 THEN K%=224
 2060 W%=W%-3:VDU K%,8,11:GOTO 2020
 2070 NEXT
 2080 ENDPROC
 2090 :
 2100 DEF FNinput(N%,C$)
 2110 L$="":*FX21
 2120 IF C$="N" s%=47:e%=58 ELSE s%=31:e
%=123
 2130 REPEAT
 2140 L%=LEN(L$):A%=GET
 2150 IF A%=13 OR A%=137 OR A%=138 GOTO
2190
 2160 IF A%=127 IF L%>0 L$=LEFT$(L$,L%-1
):GOTO 2180
 2170 IF A%>s% AND A%<e% IF L%<N% L$=L$+
CHR$(A%) ELSE A%=7
 2180 VDU A%
 2190 UNTIL A%=13 OR A%=137 OR A%=138
 2200 =L$
 2210 :
 2220 DEF PROCmenu
 2230 CLS:PROCmenus:PROCmenut
 2240 X%=10:Y%=2:*FX4,1
 2250 REPEAT
 2260 A%=GET:VDU31,X%,Y%,32,130
 2270 IF A%=138 IF Y%<5 Y%=Y%+1
 2280 IF A%=139 IF Y%>2 Y%=Y%-1
 2290 VDU31,X%,Y%,157,132
 2300 UNTIL A%=13
 2310 VDU23,1,1;0;0;0;
 2320 ENDPROC
 2330 :
 2340 DEF PROCmenus
 2350 A$=CHR$(150):B$=CHR$(34):L$="!5`"
 2360 PRINT ''A$;"  'k";SPC(31);"j`%"
 2370 PRINT A$;"j`3j";B$;
 2380 PRINT STRING$(30,"`");
 2390 PRINT "j";B$;"3k";A$;B$;L$;SPC(31)
;B$;L$;
 2400 FOR I%=1 TO 15
 2410 PRINT A$;"  5";SPC(33);A$;"5"
 2420 NEXT:L$="_05p"
 2430 PRINT A$;L$;SPC(31);L$;A$;"jpqj_";
 2440 PRINT STRING$(30,"p");
 2450 PRINT"j_qz";A$;"  tz";SPC(31);"jp4
"
 2460 ENDPROC
 2470 :
 2480 DEF PROCmenut
 2490 VDU31,27,4,-1:L$=CHR$-1
 2500 A$="~/"+L$+" ":B$=L$+"/}"+" "
 2510 PRINTTAB(12,5) A$;L$;"> ";A$;B$;B$
 2520 PRINTTAB(8,10) B$;L$;"  ~/} ";L$;
 2530 PRINT "/ ";L$;"/ ";L$;" ";L$;
 2540 PRINT">} ";A$:A$="o|"+L$:B$=L$+" "
 2550 PRINTTAB(12,6) A$;" ";L$;"  ";A$;
 2560 PRINT " ";L$;"|? ";L$;" ";L$
 2570 PRINTTAB(12,7) "||?";SPC(8);L$
 2580 PRINTTAB(12,9) L$;SPC6;L$;"  ";
 2590 PRINT L$;"  /":PRINTTAB(8,11)L$;
 2600 PRINT "|? o| o|? o| o| ";
 2610 PRINT B$;B$;B$;"o|";L$
 2620 PRINTTAB(8,12) L$;SPC22;"||?"
 2630 VDU23,1,0;0;0;0;
 2640 VDU28,5,19,36,14
 2650 PRINT SPC8; GT$;"Data input from"
 2660 PRINTTAB(10,2) CHR$157;BT$;" progr
am   ";CHR$156
 2670 PRINTTAB(10,3) GT$;"  screen    ";
CHR$156
 2680 PRINTTAB(10,4) GT$;" data file  ";
CHR$156
 2690 PRINTTAB(10,5) GT$;" text file  ";
CHR$156;
 2700  ENDPROC
 2710 :
 2720 DEF PROCscrninp
 2730 VDU26,12:PROCscrn:X%=4
 2740 FOR I%=1 TO 4
 2750 Y%=10:C%=COL%(I%):K%=279-C%
 2760 FOR J%=1 TO 12
 2770 VDU31,X%,Y%,135,157,132
 2780 D$=FNinput(4,"N")
 2790 VDU 31,X%,Y%,C%,157,K%
 2800 IF L%=0 IF A%=137 GOTO 2830
 2810 D(I%,J%)=VAL(D$):Y%=Y%+1
 2820 NEXT J%
 2830 X%=X%+9
 2840 NEXT I%
 2850 ENDPROC
 2860 :
 2870 DEF PROCscrn
 2880 A$=CHR$150+CHR$154
 2890 PRINT ',A$;"||||,,l,,|,,l,,l,<l||<
,l<,,<,,l,,l||||";
 2900 PRINT A$;"sss{ =n 1k 1j qz 1jss5j
5b  5j  1jssss";
 2910 PRINT A$;"////,,.,-.,-.,//,-.//-,.
-.,/-./,-.////"
 2920 VDU 31,29,2,-1,31,32,2,-1
 2930 PRINT'''SPC5;CHR$141;t$'SPC(5);CH
R$141;t$''SPC(3);
 2940 FOR I%=1 TO 4
 2950 C%=COL%(I%)-16
 2960 PRINT CHR$(C%);"  Item ";I%;
 2970 NEXT
 2980 PRINTTAB(0,9)
 2990 FOR I%=1 TO 12
 3000 PRINT CT$;:@%=&00000202
 3010 PRINT I%;" ";:@%=10
 3020 FOR J%=1 TO 4
 3030 C%=COL%(J%):c%=279-C%
 3040 VDU C%,157,c%,9,9,9,9,156
 3050 NEXT:NEXT
 3060 PRINT 'CT$;"     Enter number and r
eturn or use"'CT$;"    arrows to move do
wn and across";
 3070 ENDPROC
 3080 :
 3090 DEF PROCfileinp
 3100 CLS:maxf=12
 3110 DIM record$(maxf),field$(maxf),wid
th%(maxf),item%(4)
 3120 x=FNopenfile
 3130 PTR#x=0:INPUT#x,rec,recn,recs,f
 3140 FOR I=1 TO f:INPUT#x,field$,width%
(I):field$(I)=FNstrip(field$,"."):NEXT I
 3150 CLS:PRINT CT$,"Record data"'GT$;"F
ieldnames:"
 3160 FOR I=1 TO 4
 3170 item%(I)=0
 3180 PRINTTAB(12,I) CHR$148;CHR$157;CT$
;STRING$(12,".");CHR$156;
 3190 VDU31,15,I:F$=FNinput(12,"A")
 3200 IF L%=0 GOTO 3250
 3210 FOR J=1 TO f
 3220 IF F$=field$(J) item%(I)=J:GOTO 32
50
 3230 NEXT J
 3240 VDU7:GOTO3180
 3250 NEXT I
 3260 CLS:PRINT CT$,"Record data"'GT$;"S
tart record:";CT$'CT$;"   (you have ";re
c-1;" records)"
 3270 VDU31,15,1:R$=FNinput(4,"N")
 3280 n=VAL(R$):n%=rec-1:IF n<1 OR n>n%
VDU7:GOTO 3260
```

```
 3290 s%=n:e%=s%+11:IF e%>n% e%=n%
 3300 n%=1
 3310 FOR n=s% TO e%
 3320 PTR#x=256+recs*(n-1)
 3330 FOR I%=1 TO f:INPUT#x,record$(I%):
NEXT
 3340 FOR J%=1 TO 4
 3350 L%=item%(J%)
 3360  IF L%<>0 D(J%,n%)=VAL(FNstrip(rec
ord$(L%),"."))
 3370 NEXT J%:n%=n%+1:NEXT n
 3380 CLOSE#x
 3390 ENDPROC
 3400 :
 3410 DEF FNstrip(p$,c$):LOCAL I:I=LENp$
+1
 3420 REPEAT:I=I-1:UNTIL MID$(p$,I,1)<>c
$
 3430 =LEFT$(p$,I)
 3440 :
 3450 DEF PROCtextinp
 3460 CLS
 3470 x=FNopenfile
 3480 FOR I%=1 TO 4
 3490 FOR J%=1 TO 12
 3500 D(I%,J%)=FNinput1(x)
 3510 NEXT:NEXT
 3520 ENDPROC
 3530 :
 3540 DEF FNopenfile
 3550 PRINT CT$,"File data"''GT$;"     Fi
lename:";CT$'GT$;"Drive number:";CT$
 3560 REPEAT:REPEAT
 3570 PRINTTAB(16,2) STRING$(12," ");TAB
(16,3) " "
 3580 VDU31,16,2:filename$=FNinput(9,"A"
)
 3590 UNTIL filename$<>""
 3600 VDU31,16,3:dr$=FNinput(1,"N")
 3610 IF dr$>="0" AND dr$<="3" dr$=":"+d
r$+"." ELSE dr$=""
 3620 F$=dr$+filename$
 3630 x=OPENUP F$
 3640 IF x=0 PRINTTAB(1,4) CHR$129;"File
 ";F$;" not found";CHR$148;
 3650 UNTIL x>0
 3660 =x
 3670 :
 3680 DEF FNinput1(F%)
 3690 LOCAL p,p$:p$=""
 3700 REPEAT
 3710 p=BGET#F%:p$=p$+CHR$(p)
 3720 UNTIL p=13 OR p=44
 3730 =VAL(p$)
```

Introducing Laser Printers

David Spencer gives an insight into the world of laser printers.

Until very recently most people have viewed laser printers as a rather esoteric form of printer owned only by the very rich. This is hardly surprising when you consider that a laser printer could have cost at least twice as much as the rest of the computer system put together. However, laser printers are now going through the price revolution that hit disc drives about five years ago. This article will attempt to remove some of the mystery surrounding laser printers. It is not intended as a review, although we may feature a survey of available laser printers in a future issue of BEEBUG.

WHAT AND WHY?

The first question to answer is: "What is a laser printer?" The answer is simply a printer which uses a laser to produce the image on the paper, rather than the more conventional impact printing method. I will explain later exactly how the printer functions.

The second question is: "Why buy a laser printer?" There are several answers to this, though they can be summed up by saying that a laser printer is superior to a dot matrix printer in almost every respect. It is faster, quieter and produces better quality output. The only areas in which laser printers are not ahead is in the initial price and in the running costs. Both of these are several times more than even the best dot matrix systems.

MAINTENANCE

With dot matrix printers, the only maintenance needed is normally a ribbon change when the printing becomes faint, and an occasional blow out of any dust. However, the extra complexity of laser printers leads to greater maintenance needs. Cleaning is very important, and must be performed after a certain number of pages have been printed (about 1000). This normally involves cleaning all the rollers and corona wires (see later). Instead of ribbons, laser printers use a powder called *toner,* and this must be filled up regularly. Additionally, the photo-conductive master (explained later) must be changed at a certain interval. Many laser printers use changeable toner cartridges which not only include a new supply of toner, but also a new master. These can cost up to £100 each, and this explains the printing cost of several pence a page.

PAGE DESCRIPTION LANGUAGES

Most laser printers offer a similar level of intelligence to dot matrix devices. For example, there are commands to change the line spacing, the print style etc, and to enter a bit image graphics mode. Some printers, for example the Epson GQ3500, also support a set of drawing primitives which can draw shapes (circles, squares etc.) on the page at a specified position and size. There is, however, a totally different level of command interface - called a Page Description Language (PDL), of which *PostScript* is the most common.

PostScript is more than a set of commands, in fact it is a complete programming language with all the features you would expect in a more generalised language such as Basic. Also, like Basic, PostScript is an interpreted language, meaning that the program is analysed each time it is executed. Unlike a normal language, PostScript actually executes the program as it is being loaded, which in this case means transferred to the printer from the host computer. The structure of PostScript is unusual. It relies totally on the use of stacks and reverse polish notation (a type of arithmetic that operates on stacks). There is also widespread use of data types called *dictionaries,* which are lists of keywords and corresponding *objects,* where an object can be anything from a simple number to a complex procedure.

As far as built-in commands go, PostScript is, not surprisingly, rich in graphics commands.

The system works by drawing a complete image of the page directly into memory, and then printing it. Each individual graphics element is made up by forming a *path* from a series of lines and curves, and then converting that into an actual image in some way, for example by drawing the outline, or filling the path to create a solid object. Most graphics will be made up in this way, although it is possible to include bit images on the page, and these can be scaled and repositioned as necessary.

PostScript is particularly suited to printing text in a wide variety of typestyles (fonts). The definitions of the fonts are either stored in ROM within the printer, or downloaded from the host computer when they are needed. Rather than being pixel definitions, as is the screen font on the Beeb, PostScript fonts are stored as a series of lines making up the outline of each character. When a character is printed the definition is scaled to the appropriate size, and the outline filled in to produce a solid character. This has the massive advantage that whatever size the character is scaled to, the resolution to which it is printed is that of the printer. With a bit image font, resolution is lost as the characters are scaled up.

The disadvantage of PostScript is that it can add about £2000 to the price of a printer. This is partly due to the extra complexity of the printer electronics, but mainly because the rights to use the language are owned by one company, Adobe. Any manufacturer wishing to use PostScript has to pay Adobe an initial fixed sum, followed by a percentage of the cost of each printer sold.

HOW THEY WORK

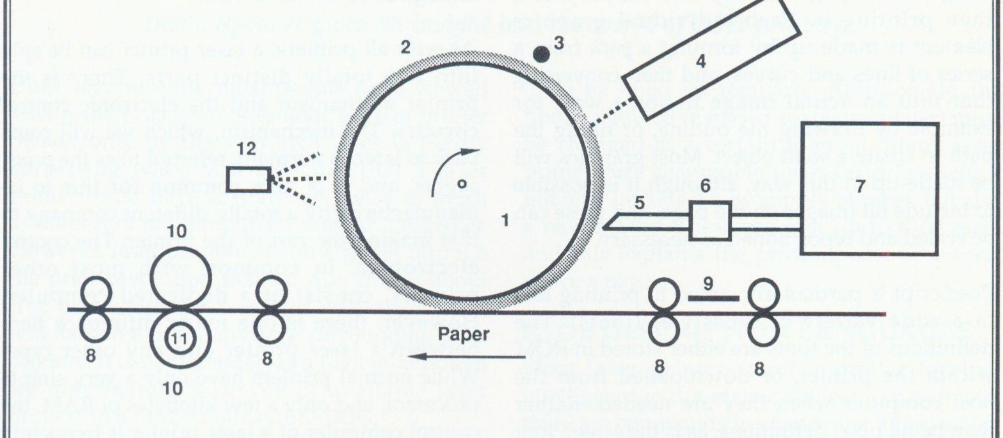
The functioning of a laser printer is totally unlike that of any other type of printer. If you've watched too many science fiction films then you can be excused for thinking that laser printers burn the image onto the page, but this is in fact far from the truth. Instead, laser printers use a Xerographic process similar to that of photocopiers. The detailed operation of all laser printers differs slightly from model to model, but the following explanation is general enough to cover all devices.

As with all printers, a laser printer can be split into two totally distinct parts. There is the printer mechanism and the electronic control circuitry. The mechanism, which we will come back to later, is normally referred to as the *print engine*, and it is very common for this to be manufactured by a totally different company to that making the rest of the printer. The control electronics, in common with most other printers, consist of a dedicated computer. However, there is one major difference here between a laser printer and any other type. While normal printers have only a very simple processor, and only a few kilobytes of RAM, the control computer of a laser printer is frequently more powerful than the computer driving the printer. For example, a fairly standard control system consists of an MC68000 16-bit processor with 1Mb of RAM and 1/2Mb of ROM. The powerful processor and large RAM are needed because the printer works at such a speed that an entire page must be built up in memory before it is printed. With a resolution of 300 dots per inch, an A4 page, assuming a margin of 1/2" on each edge, requires about 854K in which to store it.

THE PRINTER ENGINE

Figure 1 shows a schematic view of a typical laser printer engine. The Xerographic printing process works by using electrostatic charges to transfer the image onto the paper. The central component of the system is the *master*, which consists of a thin sheet of a special metallic alloy wrapped around a rotating plastic cylinder called the *drum*. This can be seen at the centre of the diagram. The master is designed so that it can hold a static electrical charge, but discharges as soon as it is exposed to light. This is achieved by constructing the master from a photoconductive material such as selenium or cadmium.

Mounted very close to the drum is a fine wire, or in some cases a mesh of wires, called the *corona wire*. By applying a very high voltage



1 Rotating drum	7 Toner reservoir
2 Photoconductive master	8 Feeder roller
3 Corona wire	9 Paper charger
4 Laser and refraction prism	10 Fuser roller
5 Toner spreader	11 Infra red heater
6 Toner charger	12 Discharger lamp

Figure 1. Simplified schematic of laser printer engine

across the corona wire, an electric field is built up, and because of the close proximity of the master, a static charge is induced on the surface of the master. In most systems this is in fact a negative electrical charge. Because the drum rotates, the entire surface of the master will be charged as it passes the corona wire.

The next stage involves the laser itself. A laser is basically a specialised form of light source. It differs from, say, a light bulb in two ways. Firstly, the light produced is *monochromatic*, which means that it is a single precise colour. Secondly, and more important to a laser printer, the light is *coherent*. This means that the light doesn't diverge after leaving the laser. In a perfect vacuum, if the ray of light leaving the laser was 1mm wide, it would still be 1mm wide thousands of miles away. In practice, air molecules deflect the beam, but it still remains coherent for several inches. In practice this means that a precise pattern can be traced out using a laser beam. In a typical laser printer, the laser beam is arranged so that it can scan the surface of the drum from left to right, rather like the electron beam within a television tube. This is performed by refracting the beam through a rotating prism, as shown in figure 2. The beam detector (a photocell) seen in the diagram is needed so that the actual position of the laser can be detected.

In a laser printer, the arrangement is such that the laser beam strikes the master just after it has been charged by the corona wire. Any area of the master which the beam strikes will be discharged, because of its photoconductive properties. By pulsing the laser on and off, it is possible to 'draw' a single line of dots on the master. A dot which will eventually be black will be a discharged point on the drum, and a blank dot will remain charged. An optical encoder disc mounted on the prism ensures that the laser is turned on and off at the correct times. Because the drum is rotating as the laser scans across, the next pass of the beam will fall on the next row down. This is very similar to a television tube, except that the surface being scanned is wrapped around a cylinder.

At this point, the image exists on the surface of the master, as a series of charges. A discharged area represents a black dot, and a negatively charged area a white dot. It is now that the actual toner is introduced. Toner is a very fine powder consisting of a mixture of a low melting point glue and carbon particles (soot). The toner is kept in a reservoir within the printer, and is fed out by a feeder system. As the toner leaves the reservoir, it is also negatively charged by a corona wire. A spreader creates a band of toner the same width as the drum. As the drum passes the toner, the electrical charges interact with each other. Because the toner is negatively charged, it will be strongly repelled by the areas of the master which are still charged. However, the toner will be attracted to the discharged regions of the master. The effect of this is to convert the image on the master from a static charge to actual toner.

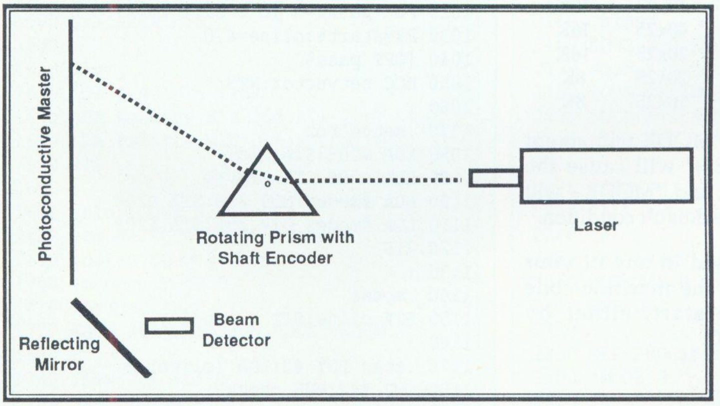


Figure 2. Optical components of a laser printer

The next part of the process involves the transfer of the toner from the drum onto the paper. Again, this uses electrical charges. As the paper enters the printer, it is given a positive charge by yet another corona wire. The drum rolls over the surface of the paper, and the negatively charged toner is attracted to the oppositely charged paper. The result is a perfectly formed image on the paper. However, at this point the toner is still loose. If you were able to pick up the paper at this point, one single blow and the toner would be everywhere. To fix the toner onto the paper a *fuser* is used. This normally consists of two very smooth ceramic rollers which are heated by an infra-red heater inside one of them. The paper is passed between the rollers which heat it up to about 200 degrees Celsius. This melts the glue in the toner and firmly fixes it to the paper.

This is not quite the end of the story. Although one page has been produced perfectly, we need to consider the next page. It is highly likely that some residual charge will remain on the master, and this needs to be removed. This can be done using either another corona wire, or more simply using a light source to remove the charge.

WHAT PRICE?

How much you pay for a laser printers depends entirely on what features you get, just as a 24-pin dot matrix printer costs more than a 3" thermal printer. What can be guaranteed is that you will pay more for a laser printer than for any other type of printer.

The cheapest laser printers are ones which perform basically the same functions as a good dot-matrix device. Examples of this type of printer are the Epson GQ3500 and the Canon LBP8-II which start at around £1300 and £1700 (inc. VAT) respectively. At the other end of the market are the PostScript-based multi-font printers such as the Apple Laser Writer II. This is available in a number of

Continued on page 60

Five New Screen Modes

Kevin Bracey's short utility provides a new star command with which you can select any one of five new modes.

This program gives you five new screen modes (modes 8-12), as described in the Advanced User Guide and BEEBUG Vol.4 No.10 p.7, but now capable of being generated by one new O.S. command: *MODE.

Type in and save the program, then run it. If all is well, you will be presented with a four colour mode 6 display telling you how to save the code. The new command *MODE is now operative.

This command works in an exactly similar same way to the Basic instruction MODE, except that like other star commands it will not accept a variable as a parameter (unless as part of an OSCLI instruction), but unlike the Basic equivalent it can be used in procedures. Giving the parameter in the range 0-7 will give you the standard screen modes, and 8-12 will result in the following:

Mode	Colours	Resolution	Text	Memory
8	16	80x256	10x32	10K
9	4	Text Only	40x25	16K
10	16	Text Only	20x25	16K
11	4	Text Only	20x25	8K
12	16	Text Only	10x25	8K

Two other parameters that *MODE will accept are + and -. Thus '*MODE +' will cause the routine to remain after Break, and '*MODE -' will cancel this effect, restoring the default condition.

The new modes may be used in any of your own programs, as long as the machine-code routine is loaded at the start, either by *ModeMC or *RUN ModeMC.

TECHNICAL NOTES

The routine is located at &900, but this may be changed if necessary, by altering line 110. It automatically checks to see if there is enough memory, and also sets HIMEM to the correct value. Basic I users will need to use the pling operator or a suitable FNequd (see BEEBUG Vol.7 No.2) to enter the values in lines 2080-2200. As checking the current mode number by looking in &355 will not return the correct value for modes 8-12, a byte has been allocated for this purpose, the location of which is displayed by the program when run.

```
   10 REM Program New Modes
   20 REM Version B2.6
   30 REM Author  Kevin Bracey
   40 REM BEEBUG  April 1989
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 150
  110 start%=&900:HIMEM=&4000
  120 PROCassemble:PROCinfo
  130 END
  140 :
  150 ON ERROR OFF
  160 MODE 7:IF ERR=17 THEN END
  170 REPORT:PRINT" at line ";ERL
  180 END
  190 :
 1000 DEF PROCassemble
 1010 IF ?&208=(start%+&1A) MOD 256 AND
?&209=(start%+&1A) DIV 256 THEN ENDPROC
 1020 FOR pass%=0 TO 2 STEP 2
 1030 P%=start%:cline=&70
 1040 [OPT pass%
 1050 BCC setvector:RTS
 1060 :
 1070 .setvector
 1080 LDA &208:STA oldv
 1090 LDA &209:STA oldv+1
 1100 LDA #modes MOD 256:STA &208
 1110 LDA #modes DIV 256:STA &209
 1120 RTS
 1130 :
 1140 .modes
 1150 STX cline:STY cline+1
 1160 :
 1170 .star LDY #0:LDA (cline),Y
 1180 CMP #42:BNE check
 1190 INC cline:JMP star
 1200 :
 1210 .ex2 JMP exit
 1220 :
 1230 .check LDY #0
 1240 .loop
 1250 LDA (cline),Y:AND #&DF
 1260 CMP cmode,Y:BNE ex2
 1270 INY:CPY #4:BNE loop
 1280 :
 1290 .space
1300 LDA (cline),Y:CMP #32
1310 BNE nospace
1320 INY:JMP space
1330 :
1340 .nospace
1350 SEC:SBC #48
1360 CMP #1:BNE onedig
1370 JSR twodig
1380 .onedig
1390 STA param:INY:LDA (cline),Y
1400 CMP #13:BEQ ok
1410 CMP #58:BEQ ex2
1420 JMP badmode
1430 :
1440 .ok
1450 LDA param:CLC:ADC #48
1460 CMP #43:BEQ in
1470 CMP #45:BEQ out
1480 LDA param:CMP #0
1490 BCC badmode
1500 CMP #13:BCS badmode
1510 CMP #8:BCS new
1520 JSR sethi
1530 LDA #22:JSR &FFEE
1540 LDA param:JSR &FFEE
1550 STA mstore:RTS
1560 :
1570 .out
1580 LDA #0:STA &287:STA &288
1590 STA &289:RTS
1600 :
1610 .in
1620 LDA #&4C:STA &287
1630 LDA #start% MOD 256:STA &288
1640 LDA #start% DIV 256:STA &289
1650 RTS
1660 :
1670 .oldv:EQUW 0
1680 .cmode:EQUS "MODE"
1690 .param:EQUB 0
1700 .mstore:EQUB 0
1710 .badmode:BRK
1720 EQUB 25:EQUS "Bad MODE":EQUB 0
1730 :
1740 .new:JSR sethi
1750 SEC:SBC #8
1760 ASL A:ASL A:ASL A
1770 STA mstore:TAY
1780 LDA #22:JSR &FFEE
1790 LDA data,Y:JSR &FFEE
1800 LDA #154:LDX data+1,Y:JSR &FFF4
1810 LDA data+2,Y:STA &30A
1820 LDA data+3,Y:STA &362
1830 LDA data+4,Y:STA &363
1840 LDA data+5,Y:STA &34F
1850 LDA data+6,Y:STA &361
1860 LDA data+7,Y:STA &360
1870 LDA #20:JSR &FFEE
1880 RTS
1890 :
1900 .exit
1910 LDX cline:LDY cline+1
1920 JMP (oldv)
1930 :
1940 .twodig
1950 INY:LDA (cline),Y
1960 CMP #13:BEQ ret
1970 SEC:SBC #48
1980 CMP #0:BCC badmode
1990 CMP #3:BCS badmode
2000 CLC:ADC #10:RTS
2010 .ret:DEY:LDA #1:RTS
2020 :
2030 .sethi:TAX:LDA himem,X
2040 CMP &3:BMI badmode:BEQ badmode
2050 LDA #0:STA &6
2060 LDA himem,X:STA &7:TXA:RTS
2070 :
2080 .data
2090 EQUD &AA09E005:EQUD &0F012055
2100 EQUD &8827D803:EQUD &03031011
2110 EQUD &AA13F403:EQUD &0F012055
2120 EQUD &8813C406:EQUD &03031011
2130 EQUD &AA09E006:EQUD &0F012055
2140 :
2150 .himem
2160 EQUB &30:EQUB &30:EQUB &30
2170 EQUB &40:EQUB &58:EQUB &58
2180 EQUB &60:EQUB &7C:EQUB &58
2190 EQUB &40:EQUB &40:EQUB &60
2200 EQUB &60
2210 ]
2220 NEXT
2230 CALL setvector
2240 *FX 247,0
2250 *FX 248,0
2260 *FX 249,0
2270 ENDPROC
2280 :
2290 DEF PROCinfo
2300 *MODE 9
2310 PRINT'"To save machine-code, type:
-"
2320 COLOUR 2:PRINT" *SAVE ModeMC ";~st
art%;" ";~P%-1;" ";~start%+3
2330 COLOUR 3:PRINT'"And to reload, typ
e:-"
2340 COLOUR 2:PRINT" *RUN ModeMC"
2350 COLOUR 3:PRINT'"Current mode is st
ored at ";:COLOUR2:PRINT"&";~start%+&B4
2360 COLOUR 3:PRINT
2370 ENDPROC
```

Understanding AC Circuits

The second program in our educational series deals with the basics of circuit theory, and is written by Keith Sumner.

Circuit theory is the study of passive electrical circuits - that is circuits consisting solely of resistors, inductors and capacitors. Taken in isolation these components are simple and uninteresting - a resistor is a thin film of carbon or similar, a capacitor consists of two metal plates sandwiching an insulating material, and an inductor is a coil of wire wrapped around an insulating core. However, connect them together and their behaviour in a circuit becomes complex to say the least.

The study of circuit theory is quite involved, requiring a good knowledge of mathematics (particularly in the fields of complex numbers, calculus and trigonometrical identities). The program here, which doesn't require a knowledge of the theory in order to use it, deals with one of the simplest areas of the subject, namely a resistor and *either* a capacitor or inductor connected in series, with a sinusoidal (AC) voltage applied across both. The applied voltage causes a current to flow through the circuit, which in turn leads to a voltage drop occurring across each component. The program given here calculates these voltage drops for a particular resistance, capacitance, inductance, and applied voltage, and also calculates the difference in phase that occurs between the two voltages. Additionally, the power dissipated by the circuit is calculated.

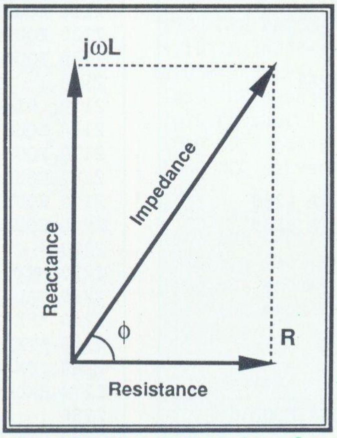


Figure 1. The construction of an impedance triangle

USING THE PROGRAM

There are two programs making up the overall system. These are given in listings 1 and 2, and should be typed in and saved as 'PHASOR1' and 'PHASOR2' respectively.

When you run the first program, you will be presented with a title page which will be displayed for a few seconds before loading the second program. You are prompted to choose between a Resistor-Capacitor (RC) circuit, or a Resistor-Inductor (RL) circuit. Next, you are asked to enter the frequency and amplitude of the applied voltage, and the resistance and either capacitance or inductance. The frequency, voltage and resistance are all entered in their base units, namely Hertz, Volts and Ohms. However, capacitances are entered in micro-Farads (one millionth of the base unit), and inductances in milli-Henrys (one thousandth of the base unit). These are the more commonly used units in electronics. So, for example, entering a value of 2 for the capacitance specifies a capacitance of $2*10^{-6}$ Farad, while a value of 0.1 for the inductance implies a real value of $1*10^{-4}$ Henrys. To experiment with, try a frequency of 1000Hz, a voltage of 10V, a resistance of 1000Ω, and either a capacitance of 0.2μF, or an inductance of 200mH.

The program then proceeds to calculate the reactance of the capacitor/inductor, and the total impedance of the circuit. Both these quantities are described later. The current flowing through the circuit is then printed along with the phase difference between it and the applied voltage. A quantity called the power factor is also printed. At the bottom of the screen you will see a schematic representation of the circuit along with the voltage drops across the two components.

Pressing the space bar will clear the screen and draw the *impedance triangle* for the circuit. This is a simple diagram showing how the impedance is the sum of the resistance and the reactance. It is in the form of an *argand* diagram which represents a complex number as a two dimensional image. The real part of the impedance (the resistance) is shown as a horizontal line, and as the resistance is always positive, this is to the right of the centre. The imaginary part of the impedance (the reactance) is shown as a vertical line. For an inductance, the reactance is positive and the line is above the centre, while for a capacitance the reactance is negative and the line is below the centre.

The white diagonal line represents the overall impedance. This is formed by drawing the parallels to the two lines to make a square, and drawing in the leading diagonal. This is shown in figure 1. The length of the diagonal is proportional to the magnitude of the impedance, and the angle between the diagonal and the horizontal (ϕ on the diagram) represents the phase shift that occurs in the circuit. This is positive if the diagonal is above the horizontal, and negative if it is below. Beginners to circuit theory often find that drawing an impedance triangle is a good way of understanding how the impedance is calculated.

A further press of the space bar will clear the screen and draw the waveforms of the voltages across the two components. These are drawn by way of phasor diagrams. A phasor is simply a line, the length of which is proportional to the magnitude of the signal, rotating around a point at a speed equal to ω. The vertical component of the line's position vector is projected onto a graph. This is best understood by observing the program in action. The yellow plot represents the voltage across the resistor, and the red plot the voltage across the inductor or capacitor. The relative magnitudes of the two voltages can be seen on the graph, as can the 90° phase shift between the two voltages.

Another press of the space bar will plot the reactive power consumption of the circuit, and a final press will plot the actual power dissipation. It is beyond the scope of this article to explain these, but the suggested reference (see below) offers a full coverage.

Finally, you are asked whether you want to re-run the program. Selecting 'Y' will allow new data to be entered, while 'N' will cause the program to exit.

THE THEORY

It would require an entire volume of BEEBUG magazines to explain circuit theory to any great depth. Therefore, what follows is only a brief coverage of the topic handled by the program. For a detailed introduction to the entire subject, I can highly recommend the book '*An Introduction to Electrical Circuit Theory*' by G. Williams, published by The Macmillan Press. This book does however assume a good knowledge of advanced mathematics (e.g. to A-level standard), including complex numbers and calculus.

The first step is to look at the representation of an AC voltage or current. Figure 2 shows two cycles of an alternating voltage. The frequency

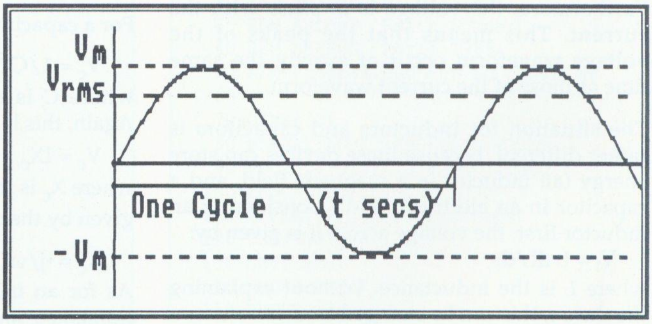


Figure 2. Two cycles of an alternating voltage

of the voltage is defined as the number of cycles per second, and the amplitude is defined as the highest instantaneous voltage reached (V_m in the diagram). You can see that in one cycle the voltage rises from zero to V_m, drops back through zero to $-V_m$, and finally rises to zero again. Mathematically, the voltage can be represented by the equation:

$$V = V_m \mathrm{SIN} \omega t$$

where ω, called the *angular velocity*, is equal to 2πf (f is the frequency). It can be very cumbersome to represent voltages in this way, and so instead, a value called the *Root Mean Square* (RMS) of the voltage is given instead. This is in effect the average voltage of the wave over any period of time, and for a sine wave it is given by the equation

$$V_{rms} = V_m/\sqrt{2}$$

The program given here assumes that RMS values are being used, and therefore if a voltage of 10 volts is specified, it actually means a sine wave that peaks at 14.14 volts.

RESISTANCE, REACTANCE AND INDUCTANCE

If an AC current flows through a resistor, an inductor or a capacitor, a particular voltage will appear across the component. For a resistor this voltage is given by a simple law called Ohm's Law, and for a current I, the voltage V is:

$$V_r = IR$$

where R is the resistance of the resistor. The value of R is constant (ignoring the effect of temperature changes), and is totally independent of the frequency of the current. Furthermore, the voltage is *in phase* with the current. This means that the peaks of the voltage waveform occur at exactly the same time as those of the current waveform.

The situation for inductors and capacitors is rather different, because these devices can store energy (an inductor in a magnetic field, and a capacitor in an electrical field). Considering an inductor first, the voltage across it is given by:

$$V_l = L\, dI/dt$$

where *L* is the inductance. Without explaining the theory, this can be changed to:

$$V_l = IX_l$$

where X_l is a quantity called the *reactance* of the inductor, and is analogous to the resistance of the resistor. The value of the reactance is given by:

$$X_l = j\omega L$$

where ω is the angular velocity of the current flowing through the inductor, and *j* is √-1. You should be able to see that the reactance, and hence the voltage across the inductor, increases as the frequency increases. Additionally, the presence of the complex value *j* means that the voltage and current are no longer in phase. Instead, the voltage is said to *lead* the current by 90°. This means that the voltage and current waves still have equal frequencies, but the peak of the voltage wave occurs at 1/4 of a cycle before the peak of the current wave. This is shown in the first two plots of figure 3.

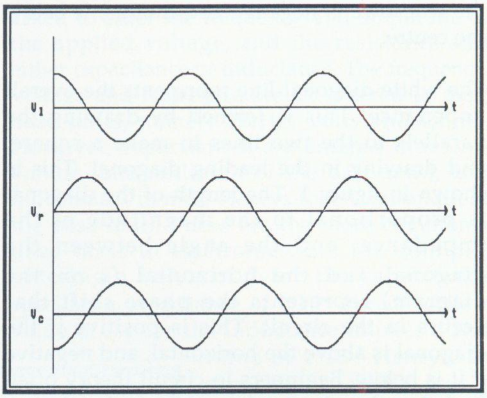


Figure 3. The phase shift between the three voltages

For a capacitor, the voltage across it is given by:

$$V_c = 1/C \int I\, dt$$

where C is the capacitance of the capacitor. Again, this leads to the equation:

$$V_c = IX_c$$

where X_c is the reactance of the capacitor.This is given by the equation:

$$X_c = -j/\omega C$$

As for an inductor, this value varies with the frequency, but in this case, because ω appears in the denominator of the equation, the voltage goes down as the frequency increases. Again, the *j* in the expression implies a 90° phase shift, but this time the voltage *lags* the current because of the minus sign. This is shown in the top and bottom plots of figure 3.

When a resistor is connected in series with either a capacitor or an inductor, the resistance and reactance can be added to give a quantity called the *impedance*, represented by the letter Z. The overall current flowing through the circuit is then the applied voltage divided by the impedence.

Next month we continue the educational series with a much simpler topic aimed directly at the teacher in the classroom.

```
   10 REM Program Phasor1
   20 REM Version B1.0
   30 REM Author  Keith Sumner
   40 REM BEEBUG  April 1989
   50 REM Program subject to copyright
   60 :
   70 PROCcode:PROCchars
   80 *TV255
   90 MODE7:PROCtitle
  100 CHAIN"PHASOR2":END
  110 :
  120 DEFPROCtitle
  130 FORX=0TO8
  140 PRINTTAB(9,X+6)CHR$141;CHR$(128+X)
"PHASOR DIAGRAMS"TAB(9,X+7)CHR$141;CHR$(
128+X)"PHASOR DIAGRAMS"
  150 NEXT
  160 PRINTTAB(3,19)CHR$129;"Phasor";CHR
$134;"Diagrams";CHR$133;"by";CHR$131;"K.
N.Sumnerá"
  170 *FX15
  180 I=INKEY(300)
  190 ENDPROC
  200 :
  210 DEFPROCcode
  220 FORP=0TO2STEP2
  230 P%=&A00
  240 [OPTP
  250 LDA#2:STA&79
  260 .prog LDY#0:LDA#3:STA&70
  270 .mainloop
  280 DEC&70:LDA #18:JSR&FFEE
  290 LDA#3:JSR&FFEE
  300 LDA&70:JSR&FFEE
  310 LDA#25:JSR&FFEE
  320 LDA#4:JSR&FFEE
  330 LDA#0:JSR&FFEE:JSR&FFEE:JSR&FFEE:J
SR&FFEE
  340 LDA#25:JSR&FFEE
  350 LDA#5:JSR&FFEE
  360 LDX#0
  370 .loop LDA &71,Y:JSR&FFEE
  380 INX:INY:CPX#4:BNE loop
  390 CPY#8:BNEmainloop
  400 DEC &79:BEQ rts
  410 LDX#2:LDY#0
  420 LDA#2:STA&70
  430 .here LDA#18:JSR&FFEE
  440 LDA#3:JSR&FFEE
  450 LDA&70:JSR&FFEE
  460 DEC&70
  470 LDA#25:JSR&FFEE
  480 LDA#4:JSR&FFEE
  490 LDA#20:JSR&FFEE:LDA#1:JSR&FFEE:LDA
#0:JSR&FFEE:JSR&FFEE
  500 LDA#25:JSR&FFEE
  510 LDA#65:JSR&FFEE
  520 LDA&7A:JSR&FFEE
  530 LDA&7B:JSR&FFEE
  540 LDA&73,Y:JSR&FFEE
  550 INY:LDA&73,Y:JSR&FFEE
  560 INY:INY:INY:DEX:BNE here
  570 JMP prog
  580 .rts RTS
  590 ]NEXT
  600 ENDPROC
  610 :
  620 DEF PROCchars
  630 VDU23,226,60,66,129,129,129,66,36,
231
  640 VDU23,225,4,60,74,137,145,82,60,32
  650 VDU23,224,128,248,30,3,3,30,248,12
8
  660 ENDPROC
```

```
   10 REM Program Phasor2
   20 REM Version B1.0
   30 REM Author  Keith Sumner
   40 REM BEEBUG  April 1989
   50 REM Program subject to copyright
   60 :
   70 DIMS%(2)
   80 REPEAT:MODE7:PROCinfo:MODE1
   90 VDU23,1;0;0;0;0:PROCcircuit
  100 PROCprint:PROCinf
  110 REPEAT:PROCdraw:PROCquest
  120 UNTILNOTrep%
  130 UNTILend%
  140 MODE7:END
  150 DEFPROCquest
  160 VDU28,0,31,19,25
  170 PRINT"Ready to plot power"
  180 PRINT"cycle of reactance ?"
  190 PROCready(0,3)
  200 PROCpower(PD,3,325)
  210 PRINT"Ready to plot power"
  220 PRINT"of combination ?  "
  230 PROCready(0,3)
  240 PROCpower(PHASE,1,325)
  250 rep%=FALSE
  260 PRINTTAB(0,0)"Enter new data and "
  270 PRINT"repeat (Y/N) ?":*FX15
  280 IF GET$<>"Y" end%=TRUE
```

```
 290 ENDPROC
 300 DEFPROCdraw
 310 VDU29,250;512;
 320 IF rep% THEN380
 330 PROCcircles(VR,VC)
 340 COLOURt
 350 PRINT"OUTER";P$;S$
 360 COLOURs
 370 PRINT'"INNER";P$;T$
 380 VDU24,265;-450;1020;400;
 390 CLG:GCOL0,3
 400 MOVE270,-250:DRAW270,250
 410 MOVE262,0:DRAW995,0
 420 VDU26,17,3
 430 PRINTTAB(15,8)"+"TAB(15,23)"-"
 440 PROCdelay(1750)
 450 VDU29,250;512;
 460 Q=4
 470 sinc=SINRAD(Q):cosc=COSRAD(Q)
 480 Sin=SINRAD(PD):Cos=COSRAD(PD)
 490 sin=0:cos=1
 500 IF rep% THEN530
 510 GCOL3,1:MOVE0,0:DRAWVR*cos,VR*sin
 520 GCOL3,2:MOVE0,0:DRAWVC*Cos,VC*Sin
 530 PROCdelay(4500)
 540 L=1:REPEAT:F=1:REPEAT
 550 VDU19,L,0;0;:PROCdelay(400)
 560 VDU19,L,L-(L=2);0;
 570 PROCdelay(1000)
 580 F=F+1:UNTILF>6
 590 PROCdelay(3500/L)
 600 L=L+1:UNTILL>2
 610 GCOL3,1:MOVE0,0:DRAWVR*cos,VR*sin
 620 GCOL3,2:MOVE0,0:DRAWVC*Cos,VC*Sin
 630 FORZ%=0TO716STEPQ
 640 stemp=sin:Stemp=Sin
 650 sin=sin*cosc+cos*sinc
 660 cos=cos*cosc-stemp*sinc
 670 Sin=Sin*cosc+Cos*sinc
 680 Cos=Cos*cosc-Stemp*sinc
 690 !&71=VC*Cos:!&73=VC*Sin
 700 !&75=VR*cos:!&77=VR*sin
 710 !&7A=Z%
 720 CALL&A00:NEXT
 730 GCOL3,1:MOVE0,0:DRAWVR*cos,VR*sin
 740 GCOL3,2:MOVE0,0:DRAWVC*Cos,VC*Sin
 750 ENDPROC
 760 DEFPROCcircles(S%(1),S%(2))
 770 sinc=SINRAD(.9):cosc=COSRAD(.9)
 780 sin=0:cos=1
 790 FORB%=0TO50
 800 stemp=sin
 810 sin=sin*cosc+cos*sinc
 820 cos=cos*cosc-stemp*sinc
 830 FORA%=1TO2
 840 GCOL0,A%
 850 X%=S%(A%)*cos:Y%=S%(A%)*sin
 860 PLOT69,X%,Y%:PLOT69,Y%,X%
 870 PLOT69,-X%,Y%:PLOT69,Y%,-X%
 880 PLOT69,X%,-Y%:PLOT69,-Y%,X%
 890 PLOT69,-X%,-Y%:PLOT69,-Y%,-X%
 900 NEXT:NEXT
 910 ENDPROC
 920 DEFPROCcircuit
 930 PRINTTAB(1,24);"~"
 940 GCOL0,2
 950 MOVE50,50:DRAW50,220
 960 MOVE50,280:DRAW50,450
 970 DRAW500,450:MOVE50,50
 980 DRAW500,50:MOVE250,50
 990 DRAW250,100:MOVE250,200
1000 DRAW250,340:MOVE250,360
1010 DRAW250,450:MOVE250,250
1020 DRAW500,250:MOVE500,50
1030 DRAW475,35:MOVE500,50
1040 DRAW475,65:MOVE500,250
1050 DRAW475,235:MOVE500,250
1060 DRAW475,265:MOVE500,450
1070 DRAW475,435:MOVE500,450
1080 DRAW475,435:MOVE500,450
1090 DRAW475,465:MOVE240,100
1100 DRAW260,100:DRAW260,200
1110 DRAW240,200:DRAW240,100
1120 IF PD=-90 THEN1180
1130 VDU5:GCOL0,0
1140 MOVE250,306:DRAW250,394
1150 GCOL0,2:MOVE250,330
1160 VDU224,8,11,224,8,11,224,4,23,1;0;
0;0;0
1170 ENDPROC
1180 MOVE225,340:DRAW275,340
1190 MOVE225,360:DRAW275,360
1200 ENDPROC
1210 DEFPROCdelay(d)
1220 FORv=0TOd
1230 NEXT
1240 ENDPROC
1250 DEFPROCready(a,b)
1260 COLOUR3
1270 PRINTTAB(a,b)"<spacebar>"
1280 *FX15
1290 REPEATUNTILGET$=" "
1300 CLS
1310 ENDPROC
1320 DEFPROCinfo
1330 rep%=FALSE:end%=FALSE
1340 P$=" CIRCLE - VOLTAGE ACROSS "
1350 CLS
1360 PRINT"You may choose one of two ci
rcuits"
1370 PRINT''"1. RC CIRCUIT"
```

Continued on page 58

1st course

Investigating Teletext Mode

Mike Williams discusses the display of text and graphics in mode 7, and in particular the use of double height characters.

Mode 7, the so-called teletext mode, is the default mode on most BBC micros. Indeed, until the advent of the Master 128 offered the opportunity for the user to reconfigure the start-up status of their machine, it was the only default mode. It certainly has its merits, notably a full choice of colours for text and graphics, yet needing only 1K of valuable memory space. Its limitations, if that's what they are, are the 40 column screen and the coarseness of teletext graphics, yet even so, good graphics are still perfectly feasible if the facilities are used with care (witness this month's feature on mode 7 histograms).

For various reasons, mode 7 is often thought of as a beginner's mode, yet it can be confusing to use, while it still has much to offer programmers of all sophistications. In this and some future articles under the First Course banner, I want to look at what can be achieved in mode 7, and to try and dispel any uncertainty that may exist about its use. In this article I shall start by considering some basic principles, before moving on to the use of double height characters as a prelude to the development of 'large' digit displays.

SOME FUNDAMENTALS

Unlike other modes, mode 7 includes a whole series of invisible control codes. These codes all have ASCII values in the same way as visible characters, but instead control the colour of text and graphics, and certain other effects on the screen. Also, although they may appear to be invisible, all these control codes occupy character positions on the screen and take up space just like other characters.

As a simple example, if you want to display the message *Hello World,* then:

```
   PRINT"Hello World"
```

will do this, and the words will start at the left-hand edge of the screen. But if we want to display the same words in friendly green this can be achieved by writing:

```
   PRINT CHR$130;"Hello World"
```

but there will be a one character gap between the edge of the screen and the start of the text message. This is occupied by the control character, in this case CHR$130. All the teletext control codes are given in the back of the User Guide for reference.

Another important point is that once a control code of any description appears on a line of the screen, it controls all the following characters on the same line until or unless another control code appears. Moreover, once a control code has been placed in position, the text or graphics can be changed without doing anything to the control codes, and the colours can be changed without touching the text.

For example, suppose (in mode 7 of course) we write:

```
   PRINTTAB(0,12)CHR$130;"Hello World"
```

to display the message as before. If we now type:

```
   PRINTTAB(1,12)"Goodbye World"
```

the original message will be replaced by the new one, which will still be in the same colour (green) because nothing has been done to change the 'green' control code at the start of the line. Now follow this by typing:

```
   PRINTTAB(0,12)CHR$129
```

and the message will stay as it is but change in colour from green to red (because we have changed the original control character). Incidentally, I do recommend you try these simple examples so that you feel clear about mode 7 text before we progress further.

We can also use teletext control codes to set a *background* colour, as well as other effects such as flashing colours (or text). Background colours are set by using the same text colour codes (range 129 to 135) in conjunction with an additional code (CHR$157) to specify that the colour is to determine a new background, and this must then be followed by a code to determine the foreground colour.

TELETEXT GRAPHICS

In principle, all that I have said about mode 7 text can be similarly applied to mode 7 graphics. There are differences, one of the most obvious being that we can type text in from the keyboard, but that this is literally impossible with regard to graphics.

Each graphics character is made up from a 2 by 3 grid of squares or pixels, the whole graphics character occupying just the same space as a normal text character. This may seem difficult to believe as graphics characters form continuous bands of colour while text characters are always separated, but this arises from the fact that text characters are carefully designed to leave a vertical line of pixels blank to provide just this separation.

As with control codes, all the graphics characters are listed in the back of the User Guide, and you may also find the teletext graphics display program published in BEEBUG Vol.7 No.8 helpful in selecting and identifying these characters.

As far as representing teletext graphics characters is concerned, we shall for the moment use the CHR$ function, together with their ASCII codes. For example:

```
PRINT CHR$146;STRING$(39,CHR$172)
```

will display a green line across the screen made up of 39 graphics characters one after the other (39 and not 40 because the control code occupies the first position on the 40 column line). The particular character (CHR$172) used here has the middle row of two pixels coloured, but the two pixels above and below this are blank.

DOUBLE HEIGHT CHARACTERS

One useful feature of mode 7 is its ability to display double height text, which can be very useful for headings and anything else where prominence or just clarity is valuable. There is a special code for this, CHR$141. This has to appear on both the top and bottom lines that make up a double height display, as does the text itself, and almost magically converts this into the correct top or bottom halves of a double height display. Here is a function which encapsulates most of the possible requirements, and demonstrates much of what I have so far described.

```
 1000 DEF PROCdouble(x,y,f,b,msg$)
 1010 LOCAL I%
 1020 FOR I%=0 TO 1
 1030 PRINTTAB(x,y+I%)CHR$141;CHR$(b);
CHR$157;CHR$(f);msg$;"  ";CHR$156
 1040 NEXT I%
 1050 ENDPROC
```

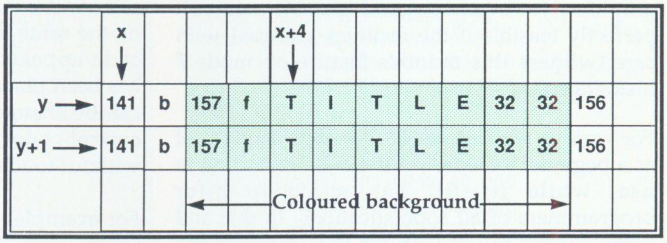


Figure 1. Mode 7 double height display

The procedure has five parameters. The first two, *x* and *y* represent a start position on the screen (which is 40 columns by 25 rows in teletext mode). Next, *f* and *b* represent the foreground and background colours as teletext codes (both in the range 129 to 135). Lastly *msg$* is the text message to be displayed. As an example:

```
    100PROCdouble(15,10,131,129,"TITLE")
```

would display *TITLE* in yellow (CHR$131) on a red background(CHR$129 for red, CHR$157 for a new background).

If you try this, you will find that the background begins two characters in front of the text, this being occupied by two control characters, and this is why two blanks (or spaces) are displayed after the text itself to provide a balanced display before returning to a default black background (CHR$156). This structure is shown in figure 1 for greater clarity.

If you use, or write your own version of such a procedure, one point you need to standardise on is the interpretation of the position given by the parameters x,y. In our definition above, the text is actually displayed four characters to the right of the specified position because of the preceding control characters. If we want x,y to be the position from which the text itself is displayed then the PRINT statement must start at (x-4,y) in order to achieve this. The choice is yours, but where several different teletext display procedures are to be written and used in the same program, it is worth standardising at the outset on one approach or the other.

DOUBLE HEIGHT DIGITS

Let us now consider the adaptation of the routine to the display of double height digits. At its simplest there is really nothing to do. Try adding the following program to the PROCdouble definition:

```
100 MODE7
110 INPUT"Time (0-999): " T%
120 REPEAT
130 T$=STR$(T%):L%=LEN(T$)
140 T$=STRING$(3-L%," ")+T$
150 PROCdouble(12,10,130,132,T$)
160 T1%=TIME:REPEAT UNTIL TIME-T1%>100
170 T%=T%-1
180 UNTIL T%<0
190 END
```

This is based on the counting programs I discussed last month, and counts down in seconds (approximately) from whatever value you input at the start. The number to be displayed is padded out with extra spaces as necessary to produce a string exactly three characters long. I have kept the coding simple, but you may wish to *tune* the program as described previously to achieve maximum accuracy.

If you don't like to see the digits displayed against a coloured background (in this case green on blue), omit all references to the background colour (parameter b) in the definition, so that line 1030 becomes:

```
1030 PRINTTAB(x,y+I%)CHR$141;CHR$(f);msg$
```

and line 150 in the program above is replaced by:

```
150 PROCdouble(16,10,130,T$)
```

Now that no coloured background has been specified, there is no need either for the extra spaces, nor the code (CHR$156) to return to a black background.

SPEED CONSIDERATIONS

If speed is of the essence, then a further refinement is possible. Even in the simplified version (with no background), two control characters are output to the screen every time a new number is displayed. If we assume that this number will have a maximum of three digits, then anything from 40% to 80% of the time taken will be used up putting the control characters on the screen. But this is quite unnecessary as we saw before.

All we need to do is to put the double height and colour codes there just once, and then just update the figures. The main program can be rewritten to achieve this:

```
100 MODE7
110 INPUT"Time (0-999): " T%
130 T$=STR$(T%):L%=LEN(T$)
140 T$=STRING$(3-L%," ")+T$
150 PROCdouble(12,10,130,132,T$)
160 REPEAT
170 T1%=TIME:REPEAT UNTIL TIME-T1%>100
180 T%=T%-1
190 T$=STR$(T%):L%=LEN(T$)
200 T$=STRING$(3-L%," ")+T$
210 PRINTTAB(16,10)T$:PRINTTAB(16,11)T$
220 UNTIL T%<1
190 END
```

The coding is longer and with some duplication, but the number of characters to be sent to the screen is considerably reduced. This does not really matter here, where we are counting in seconds, but if speed is all important, as might be the case if we were trying to monitor a signal being received through the user port or analogue port, then the saving in time would be noticeable.

As always, limitations on space force us to pause at this point. I hope I have given you plenty to think about and to experiment with. Next month I shall continue by discussing ways in which numbers can be displayed in an even larger format than double height characters can achieve.

A Screen Printer Driver for View

Bill Walker describes a printer driver for View which enables highlights to be displayed on-screen.

One of View's features is a *preview* command, called Screen, which displays text on the screen instead of printing it on paper. It is useful for checking the contents of a file, and for finding where the page breaks will occur, but it doesn't show you exactly what you will get on paper because it ignores the highlight (bold and underline) commands.

This program generates a printer driver routine (called Screen) which *prints* text onto the screen. Unlike the Screen command, this printer driver supports bold, underline and alternative character set highlights (though not extended highlights), so that "what you see" is *really* "what you get".

Type in the Basic program, and save it to disc or tape, but do *not* give it the filename Screen. When run, the program generates a printer driver routine called Screen which is automatically saved.

This printer driver can then be selected from within View by the command:

```
    Printer SCREEN
```

which loads it from disc or tape. Once loaded, the PRINT or SHEETS commands in View will now cause the text to be *printed* onto the screen (hold down Ctrl and Shift to stop it scrolling off the top of the screen). Note that you must be in screen mode 0 to 6 to see the highlights because they use user-defined graphics.

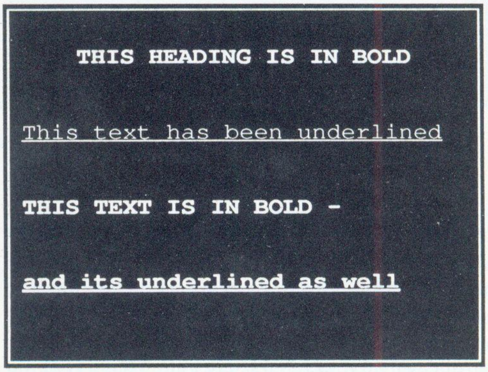
PROGRAM NOTES

The driver runs at address &400, but since this area is used by Basic it must be assembled elsewhere then relocated. This is achieved by using relative jumps throughout, and by defining the workspace labels absolutely. Thus the generator program can be run using Basic I (which does not support OPT values 4 to 7) as well as later versions of Basic.

The driver starts with a jump table of entry points, as described on page 74 of the *Into VIEW* manual. Only two of the entry points do anything - *Printer on* resets the highlight flags, and *Output character* drives the screen. The remaining entry points exit unconditionally, so that *Printer off* has no effect, and microspacing is not supported.

Highlight codes 128, 129 and 130 are trapped, and toggle the highlight flags. Other highlight codes (values>130) are ignored. Un-highlighted characters and control codes are sent to the screen directly, and are printed in the BBC's standard character set.

If the alternative character set is selected (highlight 130), then the top bit of the character is set, adding 128 to the character code. For this to be displayed, the character set must be exploded (using *FX20,6), and characters 160 through 254 defined (using VDU 23,...) before switching to View.



```
THIS HEADING IS IN BOLD

This text has been underlined

THIS TEXT IS IN BOLD -

and its underlined as well
```

Bold and underlined text

If the underline or bold highlights are selected, the character definition is read (by OSWORD 10), and after being processed, it is used to define character 255, which is then printed on the screen. The use of OSWORD to read the character, then VDU 23,255,... to define it does slow things down a little, but ensures that the driver will work with all versions of the operating system, and the 6502 second processor.

The underline highlight is generated by setting all the pixels in the character's bottom line. The bold highlight is generated by shifting the character definition to the left by one pixel, then overlaying this on the original definition. This gives vertical lines three pixels wide, instead of the usual two pixels.

The printer driver uses no workspace other than in page 4, which is reserved by View for the printer driver's use. It has been tested with View 1.4, View 2.1 and with View 3.0.

```
   10 REM SCREEN  Printer Driver
   20 REM Version B1.0
   30 REM Author Bill Walker
   40 REM BEEBUG April 1989
   50 REM Program subject to copyright
   60 :
   70 REM Highlight code 128 (default hi
ghlight 1)=Underline
   80 REM Highlight code 129 (default hi
ghlight 2)=Bold print
   90 REM Highlight code 130=alternative
 character set (symbols)
  100 :
  110 OSWRCH=&FFEE:OSASCI=&FFE3:OSWORD=&
FFF1
  120 MODE 7:HIMEM=&7B00
  130 FOR PASS=0 TO 3 STEP 3
  140 P%=HIMEM
  150 [OPT PASS
  160 \Jump table
  170 CLC:BCC CHROUT ;output char to pri
nter, supporting highlights.
  180 CLC:BCC PRON ;initialise, printer
on
  190 RTS:NOP:NOP
  200 RTS:NOP:NOP
  210 RTS ;ignore HMI
  220 :
  230 \Printer on- Initialise highlight
flags
  240 .PRON LDA #0
  250 STA FLAGS
  260 STA HIBIT
  270 RTS
  280 :
  290 \Send character to printer, proces
sing if necessary
  300 .CHROUT STX XSTORE:STY YSTORE
  310 CMP #0:BPL OCHR ;code<128=ordinary
 character
  320 \process highlight codes
  330 CMP #128:BNE HI1 ;not underline
  340 LDA FLAGS:EOR #&01:STA FLAGS:RTS ;
swap underline bit
  350 .HI1 CMP #129:BNE HI2 ;not bold
  360 LDA FLAGS:EOR #&02:STA FLAGS:RTS ;
swap bold bit
  370 .HI2 CMP #130:BNE HI3 ;not alt cha
r set so skip
  380 LDA HIBIT:EOR #&80:STA HIBIT:RTS ;
swap high bit.
  390 .HI3 RTS
  400 \Send character
  410 .SENDCH
  420 JSR OSASCI ;send char to screen
  430 LDX XSTORE:LDY YSTORE
  440 RTS
  450 \Process character
  460 .OCHR
  470 CMP #&20:BMI SENDCH ;send ctrl cod
es
  480 ORA HIBIT ;insert hi bit of code
  490 LDX FLAGS:BEQ SENDCH ;no highlight
s
  500 STA BLOCK ;build parameter block
  510 LDA #10:LDX #(BLOCK AND &FF)
  520 LDY #(BLOCK DIV &100)
  530 JSR OSWORD ;read char defn
  540 \Process underline
  550 LDA #&01
  560 BIT FLAGS:BEQ OC1 ;no uline
  570 LDA #&FF:STA BLOCK+8 ;do uline
  580 \Process boldface
  590 .OC1
  600 LDA #&02:BIT FLAGS:BEQ OC3 ;no bol
d
  610 LDX #8 ;do bold
  620 .OC2
  630 LDA BLOCK,X
  640 ASL A:ORA BLOCK,X:STA BLOCK,X
  650 DEX:BNE OC2
  660 .OC3
  670 LDA #23
  680 JSR OSWRCH ;define and print chr 2
55
  690 LDA #255:JSR OSWRCH
  700 .PRTBLK LDX #1
  710 .PRTBLK1 LDA BLOCK,X:JSR OSWRCH
  720 INX:CPX #10:BMI PRTBLK1
  730 LDX XSTORE:LDY YSTORE ;restore reg
s
  740 RTS
  750 ]
  760 FLAGS=&4F0
  770 HIBIT=&4F1
  780 BLOCK=&4F2
  790 XSTORE=&4FC
  800 YSTORE=&4FD
  810 ?&7BFB=&FF
  820 NEXT
  830 PRINT"SET UP TAPE/DISC FOR THE PRI
NTER DRIVER"
  840 PRINT"AND PRESS ANY KEY"
  850 *FX15,1
  860 A=GET
  870 *SAVE "SCREEN" 7B00 +100 400 400
  880 END
```

Page Composition for the BBC Micro (Part 2)

David James describes the programs needed to print the page layouts composed with last month's programs.

Last month, in the first part of this short series we looked at the programs needed to compose a page for printing. Now we shall consider the various programs needed to print out our pages. There are several to choose from depending on the facilities which you have available.

THE PRINTER DRIVERS

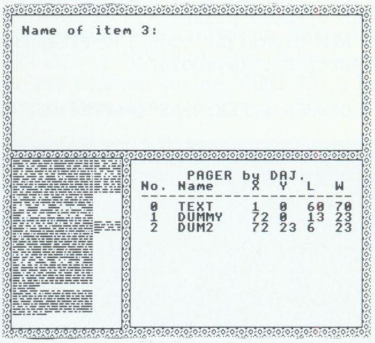
There is a choice of two different printer drivers, a standard driver (listed here) or a multi-font driver (supplied only on the magazine disc/tape). In either case your first step should be to type in, save and then run the additional program DUMPSRC, which generates the machine code file called SCDUMP. This is used by both drivers to dump the 'frame' of the page.

Of the two printer drivers the first, called simply DRIVER, is designed to work with user-generated fonts, which can be created with one of two programs to be described later. The second driver program, called DRIVERM, requires no separate font generation program, but instead works in conjunction with the PMS Multi-Font NTQ ROM. Apart from font generation, which is separate, both drivers are used and work in much the same way.

If you use the standard printer driver, you will need a font file with name 'FONTn' for each font 'n' which you have used in the page composition programs (e.g.FONT0, FONT1, etc.). The necessary font files are loaded automatically.

Either printer driver is used in the same way. Run it and then type the name of the page you wish to print out (the name you used with last month's Pager programs). The first pass draws the frame of the A4 page - if you didn't define any lines, this section is skipped.

Note that it will ask you to make sure that the print head is at the top of the page - you will have to do this at the beginning of each pass, so be sure to draw a line (in pencil) at the top of the paper and be careful to get the paper back to the exact original position each time. In my experience with a Panasonic KXP1081 printer, I have found it difficult to start printing right at the top of a piece of A4 paper due to the design of the tear-bar, so I would recommend you to use continuous stationery (i.e. fan-fold paper) to print an original, and then make as many photo-copies of this as you require. The page is complete after all the headings have been printed.



```
Name of item 3:

      PAGER by DAJ.
No. Name      X  Y  L  W
------------------------
 0  TEXT      1  0  60 70
 1  DUMMY    72  0  13 23
 2  DUM2     72 23  6  23
```

'Pager' screen dump

FONT GENERATION

Two programs are listed to generate a font for use by the standard printer driver described above when dealing with headings. The programs both generate a font called FONT1 in directory 'D' (though both name and directory can be changed as you wish). The first font generator called FGEN1 creates a 16*16 pixel font, while the much shorter FGEN3 creates a font (FONT1) based on the built-in Acorn font. Whichever you select, save the program before running it.

As they have been designed for printing headings, both font generators deal only with upper case letters (not lower case) and other symbols. For reference, each font must be 45 characters long (corresponding to characters 32 to 76 as listed in the User Guide), and each character definition must be 32 bytes long and stored in the order: top left, bottom left, top right, bottom right. Then all you have to do is load the right font - which is called "FONTn" for some n - using an OSCLI statement. This is done automatically in the standard printer driver supplied.

NOTE: We hope to feature a more versatile character definer in a future issue.

TEXT CONVERSION

Since publishing the two Pager programs last month, it has emerged that not all View files will format correctly as a result of the way in which hard and soft spaces are represented. The last program this month (just 14 lines) will convert any View text file so that it will work correctly with Pager, regardless of the column width chosen. This program is called CONV. However, still remember to omit or remove any rulers or other formatting commands from your View file before conversion.

Listing 1

```
   10 REM Program to generate fast
   20 REM screen dump needed to copy
   30 REM frame of page onto paper.
   40 REM Author David James
   50 REM BEEBUG April 1989
   60 REM Program subject to copyright
   70 :
  100 xlo=&70:xhi=&71
  110 ylo=&72:yhi=&73
  120 col=&74:co=&75
  130 times=&76:byte=&77
  140 osword=&FFF1:oswrch=&FFEE
  150 FOR I%=0 TO 3 STEP 3
  160 P%=&900
  170 [OPT I%
  180 \ fast mc screen dump
  190 JSR sendsetups
  200 LDA #&FF:STA ylo
  210 LDA #&03:STA yhi
  220 .yloop
  230 JSR sendlinesetups
  240 LDA #&00:STA xlo:STA xhi:STA co
  250 .xloop
  260 LDY co:LDA timestable,Y
  270 STA times
  280 INY:TYA:CMP #3:BNE dontreset
  290 LDA #0:.dontreset STA co
  300 JSR resetorg
  310 LDX #xlo:LDY#0
  320 LDA #9:JSR osword
  330 LDY col:LDA hitable,Y:STA byte
  340 LDX #xlo:LDY#0
  350 JSR setorg
  360 LDA #9:JSR osword
  370 LDY col:LDA lotable,Y:CLC
  380 ADC byte:LDX times:.sendloop
  390 JSR send:DEX:BPL sendloop
  400 LDA #4:CLC:ADC xlo:STA xlo
  410 LDA #0:ADC xhi:STA xhi
  420 LDA xlo:CMP #128:BNE xloop
  430 LDA xhi:CMP #4:BNE xloop
  440 LDA #13:JSR send
  450 LDA ylo:SEC:SBC #8:STA ylo
  460 LDA yhi:SBC #0:STA yhi
  470 LDA ylo:CMP #207:BNE yloop
  480 LDA yhi:BNE yloop
  490 LDA #3:JSR oswrch:RTS
  500 :
  510 .sendsetups
  520 LDA #2:JSR oswrch
  530 LDA #27:JSR send
  540 LDA #65:JSR send
  550 LDA #8:JMP send
  560 :
  570 .sendlinesetups
  580 LDA #27:JSR send
  590 LDA #76:JSR send
  600 LDA #192:JSR send
  610 LDA #3:JMP send
  620 :
  630 .setorg
  640 LDA #29:JSR oswrch:LDA #0
  650 JSR oswrch:JSR oswrch
  660 LDA #&FC:JSR oswrch
  670 LDA #&FF:JMP oswrch
  680 :
  690 .resetorg
  700 LDA #29:JSR oswrch:LDA #0
  710 JSR oswrch:JSRoswrch
  720 JSR oswrch:JMPoswrch
  730 .send
  740 PHA
  750 LDA #1:JSR oswrch
  760 PLA:JMP oswrch
  770 :
  780 .hitable EQUB 0:EQUB 240
  790 .lotable EQUB 0:EQUB 15
  800 :
  810 .timestable EQUB2:EQUB3:EQUB2
  820 ]
  830 NEXT
  840 *SAVE SCDUMP 900+100
  850 END
```

Listing 2

```
   10 REM Pager Printer Driver
   20 REM for Epson-type Printers
   30 REM Author David James
   40 REM BEEBUG April 1989
   50 REM Program subject to copyright
   60 :
  100 store%=&3800:ON ERROR GOTO 200
  110 dtable%=&1100:htable%=&1200
  120 horiz%=&1300:vert%=&1400
  130 MODE 7:PROCpageload:PROCready
  140 MODE 4:PROCoutline
  150 MODE 7:PROCtext
  160 PROCassemble
  170 MODE 0:PROCtitles
  180 MODE 7:END
  190 :
  200 MODE7:REPORT:PRINT" at line ";ERL
  210 END
  220 :
 1000 DEF PROCpageload
 1010 REPEAT INPUT "Name of page :-" P$
 1020 UNTIL P$<>""
 1030 OSCLI"L.D."+LEFT$(P$,7):ENDPROC
 1040 :
 1050 DEF PROCoutline
 1060 IF?horiz%=255AND?vert%=255 ENDPROC
 1070 FOR C%=0 TO ?horiz%
 1080 adr%=horiz%+1+3*C%
 1090 X%=adr%?0:Y%=adr%?1:L%=adr%?2
 1100 MOVE 12*X%+4,1019-12*Y%
 1110 PLOT 1,(L%-1)*12,0:NEXT
 1120 FOR C%=0 TO ?vert%
 1130 adr%=vert%+1+3*C%
 1140 X%=adr%?0:Y%=adr%?1:L%=adr%?2
 1150 MOVE 12*X%+4,1019-12*Y%
 1160 PLOT 1,0,-(L%-1)*12:NEXT
 1170 *SCDUMP
 1180 ENDPROC
 1190 :
 1200 DEF PROCtext
 1210 T%=?dtable%:IF T%=255 ENDPROC
 1220 FOR TL%=0 TO T%
 1230 adr%=store%
 1240 OSCLI"L.P."+LEFT$(P$,5)+STR$TL%+"
"+STR$~adr%
 1250 data%=1+dtable%+6*TL%
 1260 x%=data%?3:y%=data%?4:l%=data%?5
 1270 PROCready
 1280 REM Switch on printer, reset &
 1290 REM select NLQ Elite.
 1300 VDU 2,1,27,1,64,1,27,1,111
 1310 PRINTSTRING$(y%,CHR$13);
 1320 FOR L1%=1 TO l%
 1330 PRINTSPC(x%)$adr%
 1340 adr%=adr%+1+LEN$adr%:NEXTL1%
 1350 VDU3:NEXT TL%
 1360 ENDPROC
 1370 :
 1380 DEF PROCtitles
 1390 H%=?htable%:IF H%=255 ENDPROC
 1400 adr%=1+htable%
 1410 FOR HL%=0 TO H%
 1420 PROCready:CLS
 1430 VDU 2,1,27,1,64,1,27,1,77
 1440 x%=adr%?3:y%=adr%?4:f%=?adr%
 1450 h%=adr%?1:w%=adr%?2
 1460 PROCloadfont(f%)
 1470 PRINTSTRING$(y%,CHR$13);
 1480 PROCps($(adr%+5),x%,h%,w%)
 1490 adr%=adr%+6+LEN$(adr%+5)
 1500 VDU 3:NEXT HL%
 1510 ENDPROC
 1520 :
 1530 DEF PROCps(H$,x%,h%,w%)
 1540 FOR C%=0 TO LENH$-1
 1550 L%=ASCMID$(H$,1+C%,1)
 1560 IF L%>=ASC"a" THEN L%=L%-32
 1570 ch%=-1+INSTR(" !"".,:;`?0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ",CHR$L%)
 1580 ptr%=&2A00+ch%*32
 1590 PROCletter(C%,h%)
 1600 NEXT C%
 1610 FOR H1%=0 TO h%-1
 1620 PROCls(1)
 1630 Y1%=-64*H1%:PROCdump
 1640 PROCls(23)
 1650 Y1%=-4-64*H1%:PROCdump
 1660 NEXT
 1670 ENDPROC
 1680 :
 1690 DEF PROCdump W%=w%
 1700 dots=20*w%*LENH$
 1710 PRINTSTRING$(x%,CHR$1+" ");
 1720 VDU 1,27,1,90,1,dots MOD 256,1,dot
s DIV 256
 1730 FOR X%=0 TO LENH$-1
 1740 VDU 29,32*(X% MOD 32);967+Y1%-320*
(X% DIV 32);
 1750 CALL&900
 1760 NEXT X%
 1770 VDU 1,13
 1780 ENDPROC
 1790 :
 1800 DEF PROCls(L%)
 1810 VDU 1 27 1 51 1 L%
 1820 ENDPROC
 1830 :
 1840 DEF PROCletter(D%,h%)
 1850 ADR%=&3000+&1900*(D% DIV32)+16*(D%
MOD32)
 1860 let%=ADR%
 1870 PROCcolumn
 1880 let%=ADR%+8
 1890 ptr%=ptr%+16
```

```
1900 PROCcolumn:ENDPROC
1910 :
1920 DEF PROCcolumn
1930 FOR Y%=0 TO 15
1940 B%=Y%?ptr%
1950 FOR H1%=1 TO h%
1960 ?let%=B%:PROCinclet
1970 NEXT H1%
1980 NEXT Y%
1990 ENDPROC
2000 :
2010 DEF PROCinclet
2020 let%=let%+1-(632*(7=(let% AND 7)))
2030 ENDPROC
2040 :
2050 DEF PROCassemble
2060 osword=&FFF1:oswrch=&FFEE
2070 xlo=&70:xhi=&71:ylo=&72:yhi=&73
2080 col=&74:mask=&75:byte=&76
2090 FOR I%=0 TO 2 STEP 2
2100 P%=&900
2110 [OPT I%
2120 JSR pad
2130 LDA #0:STA xlo:STA xhi:STA yhi
2140 .xloop
2150 LDA #0:STA byte
2160 LDA #1:STA mask
2170 LDA #0:STA ylo
2180 .yloop
2190 LDA #9:LDX #xlo:LDY #0
2200 JSR osword
2210 LDA col:BEQ over
2220 LDA mask:CLC:ADC byte:STA byte
2230 .over ASL mask
2240 LDA ylo:CLC:ADC #8:STA ylo
2250 CMP #64:BNE yloop
2260 LDA byte:LDX &45C
2270 .outloop PHA
2280 LDA #1:JSR oswrch
2290 PLA:JSR oswrch
2300 DEX:BNE outloop
2310 INC xlo:INC xlo
2320 LDA xlo:CMP #32:BNE xloop
2330 .pad
2340 LDX &45C:.pl
2350 LDA #1:JSR oswrch
2360 LDA #0:JSR oswrch
2370 LDA #1:JSR oswrch
2380 LDA #0:JSR oswrch
2390 DEX:BNE pl
2400 RTS
2410 ]:NEXT:ENDPROC
2420 :
2430 DEF PROCc(O$,O%)
2440 VDU 1 27 1 ASCO$ 1 O%
2450 ENDPROC
2460 :
2470 DEF PROCready
2480 PRINT"Make sure that the print hea
d is at the top of the paper and press S
PACE."
2490 *FX 15,1
2500 REPEAT UNTIL GET=32
2510 ENDPROC
2520 :
2530 DEF PROCloadfont(F%)
2540 E%=OPENIN("FONT"+F%)
2550 IF E%=0 THEN F%=1
2560 CLOSE #0
2570 OSCLI"LOAD FONT"+STR$F%+" 2A00"
2580 ENDPROC
```

Listing 3

```
  10 REM Font Generator
  20 REM DATA to generate 16*16 font
  30 REM of 45 characters.
  40 REM Author David James
  50 REM BEEBUG April 1989
  60 REM Program subject to copyright
  70 :
 100 F%=&4000
 110 REPEAT
 120 READ B$:IF B$="END" GOTO 140
 130 IF ASCB$=42 PROCblock(B$) ELSE PRO
Clist(B$)
 140 UNTIL B$="END"
 150 *SAVE D.FONT1 4000+5A0
 160 END
 170 :
1000 DEF PROCblock(B$)
1010 B$=RIGHT$(B$,4)
1020 rep%=FNh(1):byte%=FNh(3)
1030 $F%=STRING$(rep%,CHR$byte%)
1040 F%=F%+rep%:ENDPROC
1050 :
1060 DEF PROClist(B$)
1070 FOR l%=0 TO (LEN(B$)-1)/2
1080 l%?F%=FNh(1+2*l%):NEXT l%
1090 F%=F%+l%:ENDPROC
1100 :
1110 DEF FNh(P%)=EVAL("&"+MID$(B$,P%,2)
)
1120 :
1130 DATA *2000,*0D01,00,*0201,*1000,*0
402,*0C00,*0440,*1900,*0203,*1E00,*0203,
06,*1500,*0201,*0200,*0201,*0A00,*0280,*
0200,*0280,*0A00,*0201,*0200,*0201,03,*0
900,*0280
1140 DATA *0200,*0280,*0500,071820,*024
0,*0280,FF,*0780,FFE01804,*0202,*0A00,FE
0F3040,*0500,*0401,00,*0201,00F00C02,*03
```

```
01,020CF0,*0380,00,*0280,00030408,*0210,
*0620
 1150 DATA *0210,080403C02010,*0208,*060
4,*0208,1020C0030509,*0C01,FF,*0F00,FE0F
3040,*0600,030C3040,*0280,FFF00C02,*0201
,020C30C0,*0600,FF0F3040,*0400,0F,*0400
 1160 DATA 8040300FF00C02,*0201,020CF00C
02,*0301,020CF0,*0400,01020408102040FF,*
0400,10305090,*0710,FF,*0410,7F,*0640,7F
,*0400,8040300FFE,*0600,F00C02,*0301,020
CF00F3040
 1170 DATA *0480,8FB0C0,*0380,40300FF00C
02,*0400,F00C02,*0301,020CF0FF,*0280,*07
00,010204081020FF,*0201,0204081020408O,*
0600,0F3040,*0280,40300F3040,*0380,40300
FF00C02
 1180 DATA *0201,020CF00C02,*0301,020CF0
0F3040,*0380,40300F,*0400,40300FF00C02,*
0301,030DF1,*0401,020CF001,*0202,*0304,*
0208,0F,*0210,*0220,*0240,E080,*0240,*03
20,*0210
 1190 DATA F0,*0208,*0204,*0202,07FF,*05
20,3F,*0820,FFF00C,*0302,0CF00C02,*0401,
020CF0071820,*0240,*0680,*0240,201807E01
804,*0202,*0600,*0202,0418E0FF,*0E20,FFE
01804
 1200 DATA *0202,*0601,*0202,0418E0FF,*0
520,3F,*0820,*02FF,*0201,*0200,*0380,*05
00,*0201,*02FF,*0520,3F,*0820,F8FF,*0201
,*0200,*0380,*0800,071820,*0240,*0680,*0
240
 1210 DATA 201807E01804,*0202,*0300,1E,*
0402,0418E0F8,*0520,3F,*0820,F81F,*0504,
FC,*0804,1FFF,*0E01,FFFE,*0E00,FE0F,*0A0
0,*0240,201807FF,*0A04,*0208,106080F8,*0
320
 1220 DATA 2122242A31,*0620,F878204080,*
0500,8040201008040FF8,*0E20,FF,*0D00,*02
01,FFE0605048444241,*0840,E007060A122242
82,*0802,07E060,*0250,48444241,*0740,E00
7,*0702
 1230 DATA 82422212,*020A,06,*0207,1820,
*0240,*0680,*0240,201807E01804,*0202,*06
01,*0202,0418E0FF,*0620,3F,*0720,*02F8,0
6,*0401,06F8,*0800,071820,*0240,*0680,*0
240
 1240 DATA 201807E01804,*0202,*0601,120A
041AE1FF,*0620,3F,*0720,F8F00C,*0402,0CF
020,*0210,*0208,*0204,*020F,3040,*0280,4
0300F,*0400,8040300FF00C02,*0400,F00C02,
*0301
 1250 DATA 020CF0FF,*0281,*0C01,07FE,*02
02,*0C00,C0E0,*0B40,*0220,18,*0207,*0B02
,*0204,18,*02E0,*0240,*0220,*0210,*0308,
*0304,*0202,0107,*0202,*0204,*0208,*0310
,*0320
```

```
 1260 DATA *0240,80E0,*0340,*0520,1011,*
0212,*0214,0807,*0302,*0504,0888,*0248,*
0228,10E0402010080402,*0201,020408102040
E007020408102040,*0280,40201008040207E04
02010080402
 1270 DATA *0801,070E040810204080,*0800,
C0FF,*0280,*0500,01020408102040,*02FF,02
040810204080,*0500,*0201,FF
 1280 DATA END
```

Listing 4

```
   10 REM Simple Font Generator
   20 REM Author David James
   30 REM BEEBUG April 1989
   40 REM Program subject to copyright
   50 :
  100 C$=" !"".,:;`?0123456789ABCDEFGHIJ
KLMNOPQRSTUVWXYZ"
  110 FOR C%=1 TO LENC$
  120 ?&70=ASCMID$(C$,C%,1)
  130 X%=&70:Y%=0:A%=10:CALL &FFF1
  140 adr%=&4000+32*(C%-1)
  150 PROCexpand(&71,adr%,128)
  160 PROCexpand(&71,adr%+16,8)
  170 PROCexpand(&74,adr%+8,128)
  180 PROCexpand(&74,adr%+24,8)
  190 NEXT C%
  200 *SAVE Z.FONT1 4000+5A0
  210 END
  220 :
 1000 DEF PROCexpand(F%,T%,M%)
 1010 FOR B%=0 TO 3
 1020 byte%=F%?B%:e%=0
 1030 IF byte%ANDM% THEN e%=e%+192
 1040 IF byte%ANDM%/2 THEN e%=e%+48
 1050 IF byte%ANDM%/4 THEN e%=e%+12
 1060 T%?(2*B%)=e%:T%?(2*B%+1)=e%
 1070 NEXT B%
 1080 ENDPROC
```

Listing 5

```
   10 REM Program to convert View files
   20 REM for use with Pager. The View
   30 REM file must be created using
   40 REM the "F" formatting option
   50 REM (CTRL-f2).
   60 MODE 7
   70 INPUT'"Name of View file :" F$
   80 V%=OPENIN("V."+F$)
   90 W%=OPENOUT("W."+F$)
  100 REPEAT C%=BGET#V%:end%=EOF#V%
  105 IF C%=&0D AND NOT end% THEN C%=&20
  110 BPUT#W%,C%
  120 UNTIL end%
  130 CLOSE #0
```

Multi-Precision Decimal Arithmetic (2)

Richard Beck concludes his machine code arithmetic routines and gives a function to greatly simplify their use.

Last month's article featured the routines to perform addition and subtraction, and gave a demonstration program that allowed you to try these out. Listing 1 given here extends the machine code to handle multiplication and division. This should be added to listing 1 from last month, making sure that the original has not been renumbered. The complete program should then be saved, and run to assemble and save the new version of 'FPcode'. The demonstration program (listing 2 from last month) can be used to try out the new extensions.

USING THE ROUTINES

Using the machine code routines in your own programs is fairly straightforward. Basically, the arguments and the operator are 'poked' into memory locations at the start of the code, and the machine code called. The result can then be read back from a different set of locations. To make all this simpler, the function in listing 2 does all the necessary conversions and calls the machine code. It then converts back the result and returns it as a string.

Before using this function, the machine code must be loaded using:

```
*LOAD FPcode 6000
HIMEM=&6000
```

This should be done in mode 7, or a shadow mode, otherwise screen memory will be overwritten. The function is then called using a command of the form:

```
ans$=FNmaths(arg1$,op$,arg2$,precison)
```

The strings *arg1$* and *arg2$* are set to the two operands. These are strings with exactly the same format as real numbers in Basic (an optional '+' or '-', followed by the mantissa, followed by an optional 'E' and an exponent). The operator is specified as the single character string *op$*, which should be one of '+', '-', '*' or '/'. The result is returned as a string in the same format as the arguments. The required precision to which the operation is to be performed is specified by the parameter *precision*.

Unlike the demonstration program, FNmaths will work with numbers that are not in standard form. In fact, the routine converts the arguments to standard form before calling the machine code. The function also works out the precision needed to hold both arguments. The result, however, will always be in standard form, although the exponent is only given if it is non-zero. For example:

```
ans$=FNmaths("-123","*","3.42E-2",8)
```

will return in ans$ the result of multiplying -123 by 3.42E-2 to 8 significant figures ("4.2066000").

If you want to call the machine code directly, for example to include the routines in a larger machine code program, then examining FNmaths will show how the values are passed to and from the routines. You might like to extend the usefulness of the routines by writing a function which can take an entire expression, for example "23*4+5.2/(2E5-1)" and convert it into a series of calls to FNmaths to obtain the overall result. This will require some mechanism to allow for operator priorities ('*' is executed before '+' etc.), but this is not too hard.

HOW IT WORKS

Performing arithmetic with floating point numbers is really not much harder than with integers. The following explanation assumes that you understand how simple integer arithmetic is performed in machine code. If not, I suggest you refer to the Exploring Assembler series published in BEEBUG. In particular, Vol.6 No.6 covered addition and subtraction, while Vol.6 No.10 looked at multiplication, and Vol.7 No.1 at division.

To simplify the routines, the little-used *decimal mode* of the 6502 is used. This changes the processor's mode of operation so that instead of a byte being treated as an 8-bit number, it is instead treated as two decimal digits. This is done by splitting the byte into two four bit nibbles, each of which contains a binary value between 0000 and 1001 which represents the value of the digit. The higher four bits represent the most significant digit. This is known as *Packed BCD* (Binary Coded Decimal). For example, the byte 01001001 can be split into the two nibbles 0100 and 1001. Therefore, this value represents the decimal number 49. The processor is switched into decimal mode using the SED instruction, and returned to binary mode using CLD. When in decimal mode, the ADD and SBC instructions treat their operands as packed BCD numbers rather than straight bytes.

The actual arithmetic is performed in much the same way that it would be done on paper. Addition and subtraction are carried out by adding or subtracting corresponding digits of the mantissa. However, before this can be done, the exponents of the two operands must be equal. If this wasn't the case, then you would end up adding or subtracting digits of differing significance. For example, imagine the sum 1.234E1+4.567. If we were to blindly add the mantissas ignoring the decimal point, then we would end up with a mantissa of 5.801 which is wrong. What we should have done was to shift one operand to equate the exponents. This would give us 1.234E1+0.4567E1. We can now add the mantissas to get the result 1.6907. The exponent of the result will be the same as that of the (shifted) operands, and therefore the actual result is 1.6907E1 which is correct.

In the above example, we equated the exponents by taking the number with the lower exponent, and shifting the decimal point left, incrementing the exponent for each shift. You might argue that we could have performed the reverse operation on the first operand, to give the sum 12.34+4.567, which gives exactly the same result. In theory the two are identical. However, in practice our routines can only store mantissas with a magnitude less than ten. In other words there must be only one digit to the left of the decimal point. Clearly, if we start shifting the mantissa left then this will no longer hold. Therefore, we are restricted to shifting the mantissa right. Any digits that 'drop off the end' (because we can only store a finite number of digits) are simply lost. They don't affect the value of the number, just the precision to which we store it. For example, if we could only store four digit mantissas, then the first example would become 1.234E1+0.456E1. We will show later how rounding can be used to minimise the effects of losing digits from the end of a mantissa.

Multiplication is no more complicated than addition. The operation is performed in two stages. Firstly, the exponents of the two operands are added to get the exponent of the result. Secondly, the mantissas are multiplied to give the mantissa of the result. The multiplication of the mantissas is done using a shift and repeated add technique. This is best illustrated with an example:

Consider multiplying the two values 4.231 and 3.912. This can be done in a number of stages:

Multiply 4.231 by 3 to give 12.693
Shift multiplicand to give 0.4231
Multiply 0.4231 by 9 to give 3.8079
Shift multiplicand to give 0.04231
Multiply 0.04231 by 1 to give 0.04231
Shift multiplicand to give 0.004231
Multiply 0.004231 by 2 to give 0.008462
Add all the results to get the answer 16.551672

What has been done is to take each digit of the multiplier, starting from the left, and multiply it by the multiplicand. This is repeated for each digit, but because the positional value of each digit is one tenth that of the previous one, the multiplicand is divided by ten at each step. The individual results, which are called the *partial products* are then summed to get the final result. In practice, a running total of the partial products is kept, rather than summing them at the end. The individual multiplications are performed by repeated addition, because the 6502 offers no multiply instruction.

Division is effectively the reverse of multiplication. The divider is subtracted from the dividend as many times as is possible without the value becoming negative. The number of times the subtraction could be performed is the first digit of the result. The remainder of the dividend is then multiplied by ten.

NORMALISATION AND ROUNDING

It is highly probable that after performing an arithmetic operation, the result will not be in standard form. The above example of multiplying the two mantissas 4.231 and 3.912 (both theirselves in standard form) gave the result 16.551672 which isn't in standard form, there being more than one digit to the left of the decimal point. Before the routine exits it must convert the number into standard form by shifting the mantissa and adjusting the exponent as necessary. This process is called *normalisation*. For example, imagine the sum:

```
4.231E-2 * 3.912E5
```

Our multiplication process will give the result:

```
16.551672E3
```

To normalise this we must shift the mantissa right one digit. However, this will divide the value by ten, so to compensate we increment the exponent by one, giving a final result in standard form of:

```
1.6551672E4
```

Another important operation to perform after the arithmetic process is rounding. We have already shown that the shifting of mantissas can result in digits being lost off the end. This results in a loss of precision in the result. If the final result of an operation requires more digits than we allow, then there is nothing that can be done about it. However, there are cases where intermediate results that occur during a calculation may need extra precision, even though the final result doesn't. To get round this, all operations are performed to an extra digit of precision. When the operation is finished, the number is rounded to the specified precision. This is done simply by looking at the extra digit. If it is less than 5, it is ignored. Otherwise, the result, excluding the extra digit, is incremented by one.

Listing 1

```
 390 CMP #ASC("*"):BNE next_op1
 400 JMP multiply
 410 .next_op1:CMP #ASC("/")
 420 BNE next_op2:JMP divide
1440 .round_loop LDA result,Y:SED:ADC #
0:CLD
1970 :
1980 .multiply LDA #0:STA nibble
1990 LDY wlenml:STY second_index:SED
2000 .loop LDY second_index
2010 LDA second,Y:PHA:AND #&F
2020 BEQ second_digit:STA pres_digit
2030 JSR add_partial_product
2040 .second_digit DEC nibble:PLA
2050 LSR A:LSR A:LSR A:LSR A
2060 BEQ next_byte:STA pres_digit
2070 JSR add_partial_product
2080 .next_byte INC nibble
2090 DEC second_index
2100 BIT second_index:BPL loop
2110 .exit CLD:CLC:LDA first+64
2120 ADC second+64:BCC mult_neg_power
2130 CMP #&7F:BCC mult_cont1
2140 JMP max_power
2150 .mult_neg_power CMP #&81
2160 BCS mult_cont1:JMP too_small
2170 .mult_cont1 SEC:ADC #&80
2180 STA result_power:LDA first+65
2190 EOR second+65:STA sign:LDA result
2200 AND #&F0:BNE no_mult_shift
2210 LDY #0:JMP left_shift
2220 .no_mult_shift JMP left_justify
2230 .add_partial_product LDY wlen
2240 LDA #0
2250 .clear_pp STA p_prod_over,Y:DEY
2260 BPL clear_pp
2270 .partial_loop LDY wlenml:CLC
2280 .inner_loop LDA first,Y
2290 ADC p_prod,Y:STA p_prod,Y:DEY
2300 BPL inner_loop:BCC no_partial_over
2310 INC p_prod_over
2320 .no_partial_over:DEC pres_digit
2330 BNE partial_loop:LDX nibble
2340 BEQ m_no_shift:JSR shift_pp
2350 .m_no_shift LDA second_index:CLD
2360 CLC:ADC wlen:TAY:SED:LDX wlen
2370 .pp_add_loop LDA result,Y
2380 ADC p_prod_over,X:STA result,Y
2390 DEY:DEX:BPL pp_add_loop:RTS
2400 .shift_pp LDY #0:LDA p_prod_over,Y
2410 ASL A:ASL A:ASL A:ASL A
2420 STA p_prod_over,Y
2430 .shift_loop INY:LDA p_prod_over,Y
2440 PHA:ASL A:ASL A:ASL A:ASL A
2450 STA p_prod_over,Y:PLA
```

```
2460 LSR A:LSR A:LSR A:LSR A:DEY
2470 ORA p_prod_over,Y
2480 STA p_prod_over,Y:INY:CPY wlen
2490 BNE shift_loop:RTS
2500 :
2510 .divide SED:LDA #0:STA remain
2520 STA result_index:LDY wlenm1
2530 .set_up_loop LDA first,Y:INY
2540 STA remain,Y:DEY:DEY
2550 BPL set_up_loop:LDA remain+1
2560 BNE div_continue:JMP max_power
2570 .div_continue STY nibble
2580 .div_loop JSR div_subtract
2590 JSR move_remainder:INC nibble
2600 JSR div_subtract
2610 JSR move_remainder
2620 INC result_index:DEC nibble
2630 LDA result_index:CMP wlent2
2640 BNE div_loop:CLD:SEC
2650 LDA second+64:SBC #&80
2660 STA result_power:LDA first+64
2670 SEC:SBC #&80:SEC:SBC result_power
2680 CLC:ADC #&80:STA result_power
2690 LDA first+65:EOR second+65
2700 STA sign:JMP left_justify
2710 .div_subtract LDY wlenm1:SEC
2720 .check_loop INY:LDA remain,Y:DEY
2730 SBC second,Y:DEY:BPL check_loop
2740 LDA remain:SBC #0
2750 BCS div_subtract_again:RTS
2760 .div_subtract_again LDY wlenm1
2770 .div_subtract_loop INY
2780 LDA remain,Y:DEY:SBC second,Y
2790 INY:STA remain,Y:DEY:DEY
2800 BPL div_subtract_loop:LDA remain
2810 SBC #0:STA remain:LDY result_index
2820 BIT nibble:BPL lower_digit_add:CLC
2830 LDA result,Y:ADC #&10:STA result,Y
2840 JMP div_subtract
2850 .lower_digit_add CLC:LDA result,Y
2860 ADC #1:STA result,Y
2870 JMP div_subtract
2880 .move_remainder LDY #0
2890 .div_move_loop:INY:LDA remain,Y
2900 PHA:LSR A:LSR A:LSR A:LSR A:DEY
2910 ORA remain,Y:STA remain,Y:INY:PLA
2920 ASL A:ASL A:ASL A:ASL A
2930 STA remain,Y:CPY wlen
2940 BNE div_move_loop:RTS
```

Listing 2

```
  10 DEF FNmaths(arg1$,op$,arg2$,pres)
  20 LOCAL sign1,sign2,exp1,exp2,count,
pow
  30 IF pres MOD 2 THEN pres=pres+1
  40 arg1$=FNnormal(arg1$)
  50 sign1=sign:exp1=exp
  60 arg2$=FNnormal(arg2$)
  70 sign2=sign:exp2=exp
  80 PROCpoke(arg1$,sign1,exp1,&6005)
  90 PROCpoke(arg2$,sign2,exp2,&6047)
 100 ?&6004=ASCop$:?&6003=pres/2
 110 CALL &6000:result=&6089
 120 res$=FNc(?result DIV 16)+"."+FNc(?
result MOD 16)
 130 IF pres=2 THEN 180
 140 FOR count=1 TO (pres-1)/2
 150 res$=res$+FNc(result?count DIV 16)
 160 res$=res$+FNc(result?count MOD 16)
 170 NEXT
 180 IF result?65>0 THEN res$="-"+res$
 190 pow=result?64-128
 200 IF pow THEN res$=res$+"E"+STR$pow
 210 =res$
 220 :
 230 DEF FNnormal(arg$)
 240 LOCAL t:sign=0
 250 IF LEFT$(arg$,1)="-" THEN sign=-1
 260 IF INSTR("+-",LEFT$(arg$,1)) THEN
arg$=MID$(arg$,1)
 270 exp=0:t=INSTR(arg$,"E")
 280 IF t THEN exp=EVAL(MID$(arg$,t+1))
:arg$=LEFT$(arg$,t-1)
 290 IF INSTR(arg$,".")=0 THEN arg$=arg
$+"."
 300 IF LEFT$(arg$,1)="." THEN arg$="0"
+arg$
 310 t=INSTR(arg$,"."):exp=exp+t-2
 320 arg$=LEFT$(arg$,t-1)+MID$(arg$,t+1
)
 330 REPEAT
 340 IF RIGHT$(arg$,1)="0" AND LENarg$>
1 THEN arg$=LEFT$(arg$,LENarg$-1)
 350 UNTIL RIGHT$(arg$,1)<>"0" OR LENar
g$=1
 360 REPEAT
 370 IF LEFT$(arg$,1)="0" AND LENarg$>1
 THEN arg$=MID$(arg$,2):exp=exp-1
 380 UNTIL LEFT$(arg$,1)<>"0" OR LENarg
$=1
 390 =arg$
 400 :
 410 DEF PROCpoke(man$,sign,exp,adr)
 420 adr?65=sign
 430 adr?64=exp+128
 440 FOR count=0 TO pres/2
 450 adr?count=VAL(LEFT$(man$,1))*16+VA
L(MID$(man$,2,1))
 460 man$=MID$(man$,3):NEXT
 470 ENDPROC
 480 :
 490 DEF FNc(a)=CHR$(48+a)
```

512 Forum

by Robin Burton

This month we'll look at discs, a topic raised frequently by your letters.

DISC COMPATIBILITY

One area which seems to cause difficulty is that of formatting discs which are to be read by other machines.

Quite a few 512 users take PC work home, and this involves transporting software from one machine to anther, and moving data in both directions via a disc format common to both systems. I do this myself, writing source programs on the 512 to be transported to a mainframe for compilation and testing, via one of a number of directly connected micros. Usually, I reverse the process at the end of the day to take the files home again for further development.

It sounds straightforward, but of course things are rarely so simple, as some of you have discovered, even though the 512 is more flexible in this area than virtually any machine I know. Most PCs can only manage a couple of formats, while cheaper machines, or those a few years old are usually limited to 360K only.

You can see some of the 512's range of disc formats on the menu of 'DISK.CMD', which you have no doubt used many times. In fact, the displayed list isn't complete, as I found by sifting through the XIOS source. Other formats include a surprising array of machines: Tandon, Philips, Zenith and others, some pretty obscure.

The range is quite impressive, and at first it appears that there is little problem. Unfortunately, in practice, discs formatted by the 512 more often than not cannot be read by PCs or clones, even though the size is correct. This applies to all 512 formats, and its cause is quite simple.

THE PROBLEM

The problem is the speed of the disc hardware in our humble Acorn micros, which operates faster than that of many PCs. Disc speed is controlled by two factors, the floppy disc controller and the type of drive used.

Consider the drives first, as they are often the ultimate limiting factor. Obviously your drives were supplied as separate items, since the BBC has none as standard. This probably means that you selected the drives, buying what you considered to be the best that you could find. No matter what you paid, your drives must be at least of a reasonable standard, or they couldn't handle the 512's 800K format.

In a PC or a clone the drives are already fitted, and clearly there is not much choice when the drives come with the machine. No prizes for guessing whether cost or performance comes first in the priorities of a manufacturer trying to build to a target price in a very competitive market.

This is one reason why many PCs (especially cheaper ones) have 360K drives. The 512 demonstrates that 800K 5.25" discs are possible while still allowing compatibility with other sizes too. This might sound cynical, but obviously there is no need for higher speed (higher cost!) drives if, as with 360K format, the data density doesn't call for it. The idea is, stick to low density formats and you needn't use the best hardware to achieve adequate reliability.

It follows that if the drives cannot manage high recording densities then the controller needn't operate quickly either. This leads to the root of the problem. The inter-sector gap written by the 512 when formatting is the difficulty; it's too small for the average PC to handle.

If you examined 512 formatted discs and compared them with real PC formats, you would find that the inter-sector gap is the only difference. I've checked discs from about half a dozen different PCs and found that the 512s are usually the smallest.

You can compare formats with a disc toolkit like the Advanced Disc Investigator. This can handle almost any track and sector size on a 5.25" disc, but to investigate inter-sector gaps you will have to write your own 6502 machine code routines.

The result of smaller inter-sector gaps is that after reading a sector, the average PC spends so long digesting the data that it misses the start of the next logical sector. The next actual sector found is therefore incorrect and the machine gives up, if you are lucky with a message like 'Bad sector ID', 'Sector not found' or something similar.

If you upgraded to a Master from a model B you may have had similar problems, in that your existing drives wouldn't work with the controller built into the Master, although they were quick enough for the 8271. The solution was faster drives, which isn't a practical option for a PC, especially if it's not yours. It is expensive, and in many cases it wouldn't solve the problem because of the controller.

You probably know that some of the variables for disc control can be altered in the BBC, by the keyboard links in a model B or *CONFIGURE FDRIVE' in a Master. If you look into the floppy disc control you'll find various factors which affect speed. One, for example, is 'head settle time'. This defines the delay the drive needs before it can read or write data, after starting up or after the heads have moved to another track.

Adjusting these settings overcomes some problems between different BBC micros, but it does not solve the PC problem.

THE SOLUTION

What is the solution? Fortunately it's simple. A disc formatted on the 512 cannot be read by most IBM compatibles, but if the disc is formatted on the PC, both machines will happily read or write to it. This of course assumes that there is a format which suits both machines; for 5.25" discs it is usually 360K or 720K.

To transfer files between your 512 and a PC, make sure that all formatting is done by the PC and not the 512. This will ensure that the discs are readable on both systems.

I have experienced this problem with several different types of PC, and this always solves the problem, if a solution exists at all. The note of caution is because sometimes there is no answer. For example, some Future PCs support a 360K format, but the 512 cannot read them, and neither can most PCs for that matter.

One other point may save someone from wasting time. Don't bother trying to read Macintosh discs. They are not PCs anyway, but that's not the problem. They use rotational speed control, which varies with the track number. Data is read or written at a constant rate, but on the outer (i.e. greater circumference) tracks of a disc more sectors are used, because the disc is made to rotate slowly. Closer to the centre, owing to the smaller physical track size, rotational speed increases and fewer sectors are used. Very few other micros do this, and discs formatted in this way can only in general be read on a computer of the same type.

HARD DISCS

One or two of you are also thinking of buying a winchester, to get more from your 512, but are unsure of how they are set up for use with DOS+.

This is a two stage process, requiring the winchester to be first formatted in ADFS mode. The format program is obviously different to that used for floppy drives, and it is usually provided along with the winchester.

During or after the formatting (which can take some time on the larger sizes), you will see messages indicating the number of errors found. There is no need to panic, a few are quite normal, given the large number of sectors and the high data density. These will be ignored when the disc is used, but if there are dozens you should insist on a replacement drive. After the formatting is complete, your winchester is ready for use as a large capacity ADFS disc.

To use the winchester in DOS+ you must then carry out stage two. This is done by booting DOS normally, then using 'HDISK', which is included on DOS+ disc 1, to create a DOS partition. This is recognised by DOS+ as drive C, and is referred to as such when you access it. If you've previously been using the winchester in ADFS mode, as with all disc operations remember the rules: before you start, ensure that you have adequate, up-to-date backups of all your files.

The size of the partition to be set up is selected during the initialisation process. This creates a locked ADFS file called DRIVE_C, of the size requested. You can also make the hard disc bootable by copying the system files to it during this process, although this is optional.

The partition size will depend on the size of your winchester and the amount of space you want in DOS, but ensure that there is enough free space (in ADFS) to allow for the DOS partition, or the initialisation will fail. The remainder of the disc, after the partition is set up, is available for use in ADFS when operating in normal BBC mode.

The DOS partition is allocated as an ADFS file, and it's a good idea to force the DOS partition to the 'end' of the disc. This is done by first *COMPACTing the existing files then *CREATEing a suitably sized dummy file in ADFS mode, to reserve the rest of the required ADFS space as a single block.

After DRIVE_C is set up, delete the dummy file, and ADFS will then be restricted to the first part of the drive, while DOS will use the second part. This results in better performance, as ADFS files won't 'jump over' the DOS partition when they extend, and unnecessary head movement is avoided. On a more unpleasant note, it also means less trouble if you ever need to recover data from the winchester the hard way, by editing sectors.

You can, of course, still boot DOS from floppies, but if you boot from the winchester there are two obvious differences. The first is that it's much quicker, and the second is that the initial screen is in white, not green as it is with floppies.

If you later decide that the partition size needs changing, you can alter it by repeating the installation process, but if you do this take note that you MUST secure all your DOS files. The vital point is that re-allocating the partition loses the previous contents, much like formatting. A new, empty root directory will always be created, because the partition may well start in a different physical place on the disc.

After creating the DOS partition all you'll have to do is copy the DOS utilities to it, and any other files you'll need. The copying operation is precisely the same as it is for floppies.

It is normal to create a directory called 'DOS' in which all the DOS utilities are put, keeping them separate from your other files. You should then set 'path' to point to this directory by the command:

```
path C:\dos
```

This command should be included in your 'AUTOEXEC.BAT' file, in the root directory of

Continued on page 43

WeightVAL Review

by Kristina Lucas

Product	**WeightVAL**
Supplier	**Nasco Software 75 Maltese Rd, Chelmsford, Essex CM1 2PB.**
Price	**£12.95 inc. VAT**

WeightVAL is a suite of programs designed to keep track of your weight, and is supplied on a 40 or 80-track disc for the BBC, B+ and the Master. It is aimed at weight-conscious individuals and health and slimming clubs, although in my view it is quite a limited package for any kind of semi-professional or professional use.

However, if what you want is to keep a record of your weekly weight changes and see a graph of your weight fluctuation during the last few years (anything between 1 and 10), you might consider adding WeightVAL to your software library. At least it has the merit of being cheap.

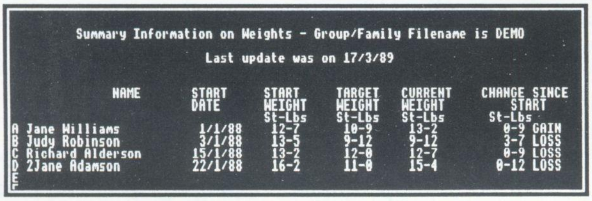
WeightVAL is a user friendly package. It is menu driven, and you do not really need to read the manual first in order to get started, although reading the manual is still advisable. The manual itself is straightforward and easy to read.

As you boot the disc you are prompted for the date. If you use a Master the current date is automatically read from the computer's clock, if not you need to enter the date manually. This date is used later when updating the records.

The main menu then appears, and from there you can choose whether you want to start a new data file, update old records or see a graphical representation of your data. Whichever option you choose to explore the package, it is easy to return to the main menu at any time by pressing the Escape key.

A demonstration group of files is provided for you to examine, change records and find out how the package works. Unfortunately this demo is offered as a default, so when you want to access your own files, you have to select that option.

You can create your own group of files by going to the utilities menu. Each file relates to an individual and contains their weight records: current weight, starting weight and date, target weight and weight change since the last record. A group can consist of up to 15 individual files. The records are entered in tabular format, and a line is allocated for each individual. The software provides an edit option, which allows you to amend the records of a specific individual, or an update option, where you can change the records of a whole group of individuals.

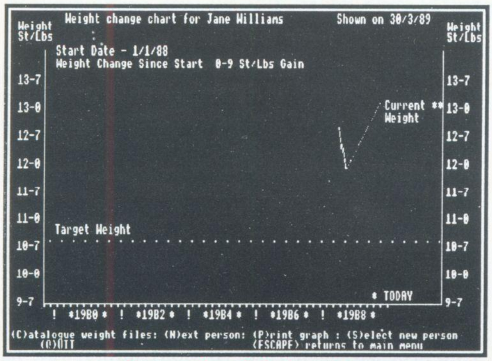


```
     Summary Information on Weights - Group/Family Filename is DEMO

                     Last update was on 17/3/89

          NAME      START    START   TARGET  CURRENT   CHANGE SINCE
                    DATE     WEIGHT  WEIGHT  WEIGHT       START
                             St-Lbs  St-Lbs  St-Lbs     St-Lbs
A Jane Williams     1/1/88   12-7    10-9    13-2         0-9 GAIN
B Judy Robinson     3/1/88   13-5    9-12    9-12         3-7 LOSS
C Richard Alderson 15/1/88   13-2    12-0    12-7         0-9 LOSS
D 2Jane Adamson    22/1/88   16-2    11-0    15-4        0-12 LOSS
E
F
```

Summary weight table

Weight records can be entered in stones, pounds or kilograms (you can choose the units in the appropriate menu), and updated once weekly. For some reason the program will not allow you to amend your records more frequently, and if you try to do that, the last record can be overwritten. This is puzzling, as I believe that anyone who is so keen to monitor their weight on a computer, would like to have greater precision in their records, especially in the case of a health centre.

One thing that puzzled me even further is that you have to suggest to the computer your target weight: the package does not provide any recommendations as to what your ideal weight should be (a very simple thing indeed). It would be nice if the software incorporated reference tables, or a way of calculating your ideal weight. It could even suggest how much weight you need to lose or gain each week in order to achieve a certain target, and that could be based on health recommendations.



```
Weight change chart for Jane Williams          Shown on 30/3/89
Start Date - 1/1/88
Weight Change Since Start  0-9 St/Lbs Gain
(C)atalogue weight files: (N)ext person: (P)rint graph : (S)elect new person
     (Q)UIT                               (ESCAPE) returns to main menu
```

Personal weight graph

As it is, this package is little more than an electronic notebook for recording your weight on a weekly basis. The one thing it will do for you is tell you whether you have gained or lost weight in the last week - yippee! And it will draw a graph of your weight fluctuations for any period of one to ten years.

So long as you have a few week's records in store (remember that you can backdate your entries by typing the date of your choice at the beginning, that is if you know how much you weighed two or three weeks ago), you can have your weight chart drawn. The chart shows your target weight, your initial weight, starting date and the changes up to the present time. I couldn't work out how the starting point for the grid was chosen; for instance one of the charts I drew started at 10 stone, while all the records I entered were below 9 stone, so I ended up with an invisible graph.

If you need a printed record, that is very easy indeed. A print option is provided on both the summary weight table and the weight chart screens.

To summarise, WeightVAL is an inexpensive user-friendly package for keeping your weight records. If that's what you want to do, WeightVAL might be the package for you. However, if you set out with greater expectations, you might be disappointed.

512 Forum (continued from page 41)

drive C. This ensures it is always issued when booting the system from the hard disc.

DISC TIPS

If you find that none of the partition sizes in the standard hard disc set-up suits you, you can '*CREATE DRIVE_C' yourself in ADFS mode, to any size you like. Naturally the comments on backups apply equally if you use this approach.

After creating 'DRIVE_C' don't forget to lock it, because that's all that prevents ADFS from being able to delete it. Next, boot DOS+ from floppies and proceed as before, setting up directories and copying files as necessary.

Finally, did you know that there is a much quicker way to format 800K discs than using the laborious 'DISK' command. On the Gem Applications disc (No.2), you'll find a directory 'GEMSYS' and in it a file called 'FORMAT.COM'. Copy this to your DOS utility disc. Then, to format an 800K disc just type 'FORMAT'.

There is no menu, but you can append a drive identifier to indicate the drive to be used. Assuming A: as the current drive, with the program on the disc in A:, to format drive B: the command would be:

```
format b:
```

If you omit the drive identifier formatting takes place on the current drive. This is quite safe, as the program always prompts for a new disc in the appropriate drive, then waits for a key-press before proceeding.

It's more convenient than 'DISK' and as it doesn't verify it is much quicker too - but beware! If you tell it to format an already formatted disc, it does NOT stop to request confirmation.

File Handling For All (Part 10)

David Spencer and Mike Williams take a look at sorting data files.

SORTING

A fundamental requirement of a database system is the ability to access a set of records in a recognisable order. This is of particular importance when a set of records are printed out for reference. For example, to find a particular name among thousands is easy if the list is in alphabetical order, but considerably more time-consuming if it is unordered.

KEY FIELDS

As with searching, it is not normally necessary to examine all the fields in a record when sorting a file. For example, in the simplest case you might just wish to sort all the records so that just one field is in order. In this case, only that field needs to be examined. In a more complicated example, you might wish to sort a file into alphabetical order of surname, and for multiple occurrences of the same surname, sort them into order of christian name. If the two parts of the name were stored in separate fields, then both would need to be examined.

We can use the idea of a key field for sorting, just as we did for searching. A template record can be set up to specify which fields in each record should be compared when sorting them. By adopting this system, we could change the basis of the sort just by changing the key field template. For example, we could specify the name field as the key to sort the records into name order, or, say, a date field to sort the records chronologically.

THE COMPARISON FUNCTION

The use of key fields for sorting raises another question. Obviously the sorting process will require the comparison of key fields to decide the order they should be in. In many cases the ordering will be obvious. For example, when putting fields into alphabetical order the string comparison functions built into Basic can be used to compare strings. But suppose we are sorting date fields. Even if the dates are stored as strings, it is not sufficient to compare them directly using Basic functions. This is because Basic's comparison functions can handle strings or numbers, but not dates directly. What we need is a function that can compare two dates, and decide which one should come first. The operation of such a function, called a *comparison function*, will depend on the exact format that our dates are stored in. In general, we will need an individual comparison function for each type of field which can be used as a key in a sort.

INTERNAL AND EXTERNAL SORTING

There are broadly speaking two ways of performing a sort operation. The first, known as *internal sorting* is characterised by all the records that are being sorted residing in memory at once. On the other hand, if there is insufficient memory to hold all the records, then *external sorting* must be used, and effectively sorts a file into order by loading in just a part at a time. External sorting offers the advantage of sorting data files many times larger than the available memory. We will start, though, by looking at internal sorting methods.

ALL SORTS OF SORT

There are many different ways of actually sorting a list of records. Each of these methods offers its own set of advantages, and it normally follows that the faster the sort, the longer and more complex the sort routine required. All the sorting routines share one common attribute - they work by performing a series of comparisons between the key fields of two records, and depending on the result of the comparison swapping the records over. We will look at a number of sorting algorithms here. To simplify our study, we will assume that in each case we are sorting an array of *n* numbers into ascending numeric order. It should, however, be easy to see how to change the routines to sort files of records.

THE SELECTION SORT

Conceptually, this is probably the simplest sort to understand, and the one which most people would end up using if asked to design a sorting routine out of the blue. The purpose of our sort is to end up with the lowest number first, then the next one, and so on. So, all we need to do is to search the entire list for the lowest value and swap that with the number at the start of the list. We then repeat the process ignoring the first value in the list, and so on until we get down to just one single number left in the list. A routine to sort an array called *data* using this method could be:

```
1000 DEF PROCselection(n)
1010 FOR count = 1 TO n-1
1020 index=count:min=data(index)
1030 FOR i = count+1 TO n
1040 IF data(i)<min THEN index=i:
     min=data(index)
1050 NEXT i
1060 data(index)=data(count)
1070 data(count)=min
1080 NEXT count
1090 ENDPROC
```

This should be largely self explanatory. Lines 1060 and 1070 perform the swapping of elements, and use the fact that *min* already holds the value of *data(index)* to avoid needing a temporary variable.

While the selection sort is very simple to understand, and easy to implement, it is not that efficient for a number of reasons. In particular, the number of comparisons that are made during the entire sort is much greater than that of other methods. The time taken for the sort is also roughly proportional to the square of the number of items. Therefore, doubling the length of the list will quadruple the time taken, and increasing the list length by a factor of ten will increase the time a hundred fold.

THE BUBBLE SORT

An improved sorting method is the bubble sort, the method which most people learn about first. As its name suggests, this method works by 'bubbling' the higher values to the end of the list. The bubble sort is actually performed by comparing adjacent numbers in the list (starting with the first and second, then the second and third, and so on), and swapping them if they are the wrong way round. This is repeated until the penultimate number has been compared with the last one in the list. At this point, the last number in the list will be the highest value, and is therefore in the correct place. Figure 1 shows this for just five numbers. The bubbling process can then be repeated ignoring the last number, which we know is already correct. This will put the second highest value in the correct place. We could continue this for a total of *n-1* times, however, there is a short cut. If in any pass no numbers are swapped, then the list is already in order. We can therefore keep a flag which is cleared at the start of each pass, and set if a swap is made. If the flag is still clear at the end of a pass, then the sort is complete. The following is a simple bubble sort routine:

```
1000 DEF PROCbubble(n)
1010 maxindex=n-1
1020 REPEAT
1030 flag=FALSE
1040 FOR count = 1 TO maxindex
1050 IF data(count)>data(count+1) THEN
     temp=data(count):data(count)=
     data(count+1):data(count+1)=temp:
     flag=TRUE
1060 NEXT count
1070 maxindex=maxindex-1
1080 UNTIL maxindex=1 OR NOT flag
1090 ENDPROC
```

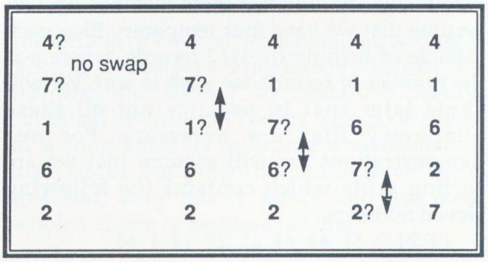


Figure 1. One pass of a bubble sort showing how the largest valve moves to the end.

The bubble sort is better in terms of execution time than the selection sort (especially when a new record is added to an otherwise sorted list, in which case only one pass is needed), but it still suffers from the problem of escalating time as the number of items is increased. To get around this we need to move towards more complicated sorts, such as the quick sort.

THE QUICK SORT

This is a sorting method which works in a totally different way to the bubble or selection sorts. It works by taking a value from near the centre of the list of records, and by a series of swaps ensuring that all records before the chosen one are less than it, and all ones after it are greater than it. The list is then split into two at the chosen record, and the process repeated for the two new lists. This is continued recursively until all the new lists contain just one record, at which point the complete list will be in order.

The quick sort is too complex to give a detailed description of here. Instead, you are referred to either of the books mentioned later, both of which give a full description.

EXTERNAL SORTING

When it comes to performing an external sort, there is no longer the proliferation of different routines which exist for internal sorts. Furthermore, all the methods work on the same basic principle - the merging of a number of lists, sorting them in the process. We will consider two different methods here - The balanced two- way merge, and the natural two-way merge. To illustrate these methods we will assume that we have four temporary files, each capable of holding (n+1)/2 records, where *n* is the number of records we wish to sort. We will show later that in practice not all these temporary files are necessary. For our demonstrations we will assume that we are sorting a file which contains the following eleven numbers:

```
19 24 3 51 43 64 21 29 34 4 48
```

THE BALANCED TWO-WAY MERGE

Let us take the data from our file, and split it into two nearly equal lists which we store in temporary files A and B:

```
A = 19 24  3 51 43
B = 64 21 29 34  4 48
```

Now, take the first element from each file and write them in order as a pair to file C. Take the next element from A and B, and write them in order to file D. Repeat this, alternating between files C and D until all the elements are used up. This gives:

```
C = 19,64  3,29  4,43
D = 21,24  34,51  48
```

The next step is to merge corresponding pairs from C and D into quadruplets in A and B. This is done by taking the first pair from C, and the first from D. We then compare the first number of each (19 and 21), and output the lower one to A. The value sent to A is replaced with the next one in its pair (64 in this case), and the comparison repeated. This continues until one pair is exhausted, when the remainder of the other pair is sent directly to A. Working this through results in the quadruple 19,21,24,64 being output to A. The entire process is repeated for the second pairs, sending the resulting quadruple to file B. The third pairs are merged into file A, and so on. This gives:

```
A = 19,21,24,64   4,43,48
B =  3,29,34,51
```

The next stage is to merge the quadruplets into groups of eight number in C and D, using exactly the same technique. This leads to:

```
C = 3,19,21,24,29,34,51,64
D = 4,43,48
```

This continues, merging two groups together, until all the values are in a single file. In our example this requires just one more pass, resulting in:

```
A = 3,4,19,21,24,29,34,43,48,51,64
B = <empty>
```

This merging process will work with files of any length, although unless the length of the initial file is a power of two, the lists quickly cease to be of equal length (as is the case in our example).

THE NATURAL TWO-WAY MERGE

While the balanced two-way merge is easy to follow, it is not the most efficient merging method. An improvement is offered by the natural two-way merge. This starts by splitting the data into lists A and B as before. It then proceeds by taking the first numbers from A and B, comparing them and sending the smaller of the two to file C. The value sent is replaced by the next one from its file (as in the balanced merge), and this is repeated until the smaller of the two values is less than the last value sent to C. If we started with A and B as before, at this point C would contain the numbers: 19 24. The comparison is then continued, but now sending the larger of the two values to C. Again, this continues until the larger value is smaller than the one previous sent. This gives: 19 24 64 in C.

We now repeat the entire process from the start, but this time sending the output to file D. We then do it again for C, and so on until both A and B are exhausted. This will result in:

```
C = 19 24 64  4 43 48
D =  3 21 29 34 51
```

(We suggest you try this through on paper to see how it actually progresses).

We now repeat the entire merging process, taking the numbers from files C and D, and putting them back to A and B. This leaves the situation:

```
 A =  3 19 21 24 29 34 51 64
 B =  4 43 48
```

This is continued until all the values are contained within just one file (either A or C), in which case this file contains the sorted list.

TEMPORARY FILES

In practice we do not need four temporary files. Instead, we can get away with just one additional file, the same length as our original data file. The two files (the temporary one and the original) can then be split into two by keeping two pointers to them and switching them using the PTR# statement. All the merging is then performed from one file to the other, and then back again, and so on.

IMPROVEMENTS

So far we have considered internal and external sorts as totally separate. However, it should be apparent from the description of the merging methods that any number of pre-sorted files can be merged together to produce a single ordered file. Therefore, we could sort a large data file by splitting it into chunks that would fit into memory, sorting each of these with a fast internal sort, and then merging the sorted chunks together to form a final file. This would be *much* quicker than using a merging method to sort the entire file.

Another major improvement follows from our earlier statement that it is only necessary to consider certain fields during the sort. It is therefore only necessary to read into memory the relevant fields, and a reference to the complete record (its existing pointer say). Using this method, rather than sorting whole records, you sort the key fields (and pointers). When the sort is complete, you are left with an sorted list of record numbers. It is then sufficient to read the complete records from the original file in the correct order, and write them to a new file.

MORE SORTS

There have been numerous articles and books written about sorting methods. A particularly good book is called *'Information Representation and Manipulation using Pascal'* by E.S. Page & L.B. Wilson, published by the Cambridge University Press. This book covers a lot of material, and is fairly easy to read. Incidentally, despite its title, it doesn't require any knowledge of Pascal to understand most of the content.

The definitive book on searching and sorting is *'The Art of Computer Programming Volume 3 - Searching and Sorting'* by Donald E. Knuth, published by Addison Wesley. This book covers all sorting methods including a mathematical analysis of the performance of each. It is, however, very heavy going and not most people's choice of bedtime reading.

CONCLUSION

Hopefully the theory contained in this article will enable you to incorporate sorting routines into your own file handling programs. The routines given for the internal sorts can be directly modified for your own use, while the natural merge technique described here could be modified and used as the basis for an external sorting program.

Spin a Disc

David Spencer gives the technical low down on discs.

A revolution in disc drives over the last five years has ensured that almost all Beeb users will own a disc drive of some form or the other. However, as with much high technology, many people treat the disc drive as a 'black box' which is used to store data, never stopping to think about how it works. In this article I will explain the operation of a disc drive from the lowest level up.

DISC ORGANISATION

When you take a brand new disc out of the box (we will assume it is a 5.25" one), then it is totally blank. All you have is circular disc inside a square envelope. The disc and envelope have a large hole in the centre for the drive spindle to pass through, and the envelope has a slot either side for the read/write head to touch the disc. Additionally, the envelope has a write protect notch and a hole near the centre which lines up with a small hole near the centre of the actual disc, called the *index hole*. There are also two notches, one either side of the slot, and these are used to align the disc within the drive.

The actual disc is coated on both sides with a magnetic material similar to that used on cassette tapes. (All discs are made double sided - it is the testing process which grades them.) Initially, this will be arranged randomly, and for all intents and purposes no data is recorded on the disc.

When data is stored on a disc it is arranged in a number of equally spaced concentric circles called *tracks*. On modern disc drives there will be forty or eighty such tracks, and these are numbered starting at zero, with track 0 being the outermost one on the disc. There are no physical markings on the disc to separate tracks, instead, the positioning of the tracks is determined by the movement of the read/write head. This is on a slider which can be moved across the surface of the disc by a stepper motor or similar. The head will have forty, or eighty, distinct positions that it can be in, and it is this which dictates the track layout. Incidentally, when the disc controller in the computer moves the head, it does so by issuing two signals to the disc drive. One tells the head to move by one track, and the other tells the head in which direction to move (in or out). If the controller wishes to move to a specific track, it needs to know which track the head is currently on. To achieve this, the drive returns a signal to the controller whenever the head is on track zero. The controller can then keep track of the current position as the head is moved.

The only other position sensor built into the drive is the index hole switch. This sends a signal to the controller each time the index hole in the disc lines up with the corresponding hole in the envelope. In other words, the signal is sent once for each rotation of the disc. Some types of disc have many index holes in them. These are called hard-sectored, and are not suitable for use with the Beeb.

RECORDING FORMATS

When data is recorded onto a cassette tape it is converted into audio tones before being recorded. There are two tones, one for a logic 0, and the other for a logic 1. This is because the circuitry of a cassette recorder is designed to handle audio signals and not digital data. Disc drives don't suffer from this restriction and they are able to record the digital data directly, thereby achieving much higher storage densities. However, it is not adequate simply to store the bits of data directly on the disc. If this was done then the data would need to be read back at *exactly* the same rate as it was written. If a variation in drive speed of even 0.1% occurred, then the data would be corrupted when re-read.

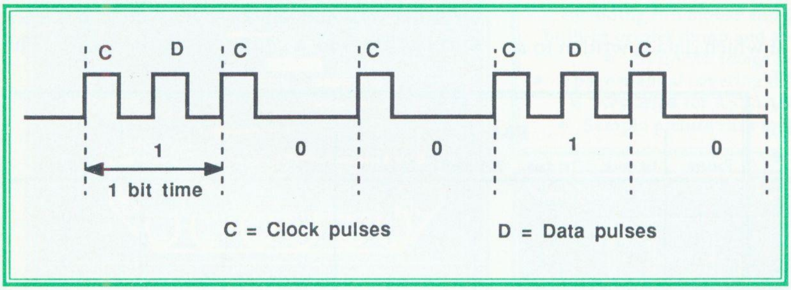


Figure 1. Data format for the valve 10010 stored using FM

To get around this problem, two bits are recorded on the disc for each bit of data. The first bit is always a logic 1, and is called the *clock bit*. The second is the actual data bit. The bit pattern written to disc for the bits 10010 is shown in figure 1. The disc controller can synchronise itself to the clock pulses, and in this way cope with speed variations of up to several percent. This recording system is known as *Frequency Modulation* (FM).

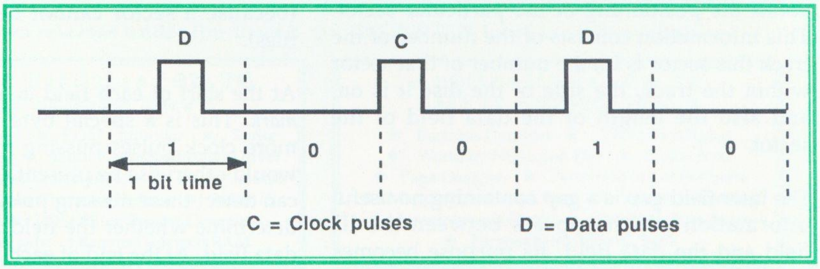


Figure 2. Data format for the valve 10010 stored using MFM

While FM provides a solu-tion, it is wasteful in that a clock bit is required for each bit. There is, however, a simple way of reducing this requirement. Referring to figure 1, it can be seen that when the data bits are ones, they could be used to synchronise the reading. With this is mind, it is only necessary to record a clock bit if the data for this bit is zero and the data for the last bit was also zero. Figure 2 shows this for the pattern 10010 again. You will see that if the clock bit is missing, then at least one of the previous or current data bits will be a one, and this can be used for synchronisation. This recording method is called *Modified Frequency Modulation* (MFM).

To see the advantage of MFM over FM, consider the number of bits written to the disc for the four possible combinations of two bit (00, 01, 10 and 11). Table 1 shows the clock and data bits recorded for each pair in both FM and MFM. When considering the density of stored data, it is the pulses corresponding to logic ones which matter. For any recording medium there is an upper limit to the number of pulses that can be recorded on a given area of the medium.

This limit is determined by the quality of the magnetic material used in the manufacture of the disc. If we could reduce the average number of pulses per bit of data, we could fit more data on a disc. Looking at table 1, to record eight bits (four two-bit combinations) using FM requires 12 pulses. This is an average of 1.5 pulses per bit (as would be expected, because each bit requires a clock pulse, and there is a 50/50 chance of it having a data pulse). Using MFM recording, the eight bits require a total of five pulses (four data and one clock), giving an average of 0.625 pulses per bit. Therefore by using MFM as opposed to FM, the amount of data recorded can easily be doubled. As you might guess, on the Beeb DFS uses FM recording, and ADFS uses MFM recording.

DISC FORMATS

Because of the speed at which data is written to a disc (125000 bits per second with FM, 250000 with MFM), data has to be organised on the disc in chunks, which are accessed as a simple unit. These chunks are called sectors. Before a disc can be used to store data, it must have these sectors laid out of each track - a process called *formatting*.

Each sector is made up of three distinct parts - the ID field, the inter field gap, and the data field. The ID (identity) field holds information about the positioning of the particular sector. This information consists of the number of the track this sector is in, the number of that sector within the track, the side of the disc it is on, and also the length of the data field of the sector.

The inter field gap is a gap containing no useful information which appears between the ID field and the data field. Its purpose becomes apparent when a sector is written to the disc. Once a disc has been formatted, the ID fields are never rewritten, unless it is reformatted. Instead, when a sector of data is written, the disc controller steps to the correct track, and then starts reading ID fields on that track until the correct sector is found. The controller then rewrites the data field of the desired sector. The process of switching from read mode to write mode is not instantaneous, and therefore the gap is necessary to allow the switch over to occur between reading the ID field and writing the data field.

The data field holds the data for that sector. The length of each data field is determined by a byte in the ID field, but it is normally restricted to one of 128, 256, 512 or 1024 bytes. A bigger sector size means that more data can be fitted onto the disc because less storage capacity is wasted in the ID fields. For example, the ADFS L format uses sixteen 256-byte sectors per track and achieves a capacity of 640K. The Archimedes ADFS D format uses five 1024-byte sectors per track and manages 800K a disc. However, because sectors have to be written as complete units, a large sector size means that space is wasted when small files are used (because a sector cannot be split across two files).

	FM				MFM			
Pattern	1st clock	1st data	2nd clock	2nd data	1st clock	1st data	2nd clock	2nd data
0 0	1	0	1	0	0	0	1	0
0 1	1	0	1	1	0	0	0	1
1 0	1	1	1	0	0	1	0	0
1 1	1	1	1	1	0	1	0	1

Table 1

At the start of each field is a so-called *address mark*. This is a special byte which has one or more clock pulses missing from it, when they would otherwise be present. The disc controller can detect these missing pulses and use them to determine whether the field is an ID field or a data field. At the end of each field there is a two byte error check called the CRC (Cyclic Redundancy Check) code. This is a special checksum performed on all the data in the field, and is used by the disc controller to check for read errors.

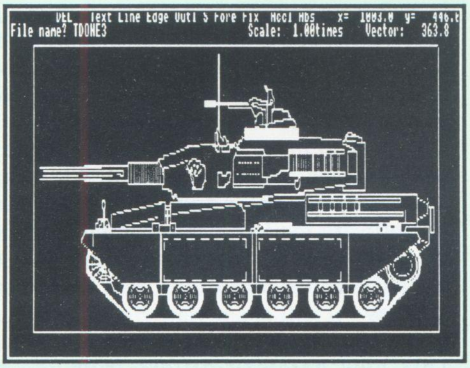
The exact layout of each sector depends on the format in use (DFS, ADFS etc.). Basically, DFS uses ten sectors of 256 bytes, and ADFS uses sixteen sectors of 256 bytes per track.

Continued on page 60

ASTAAD

The Best of BEEBUG

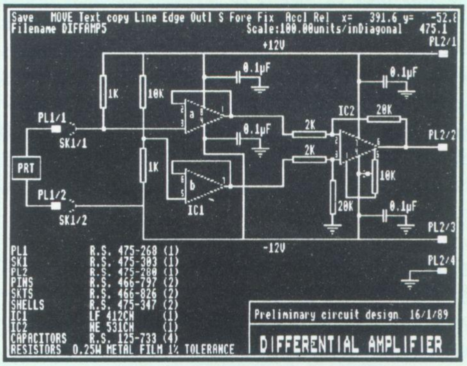
Our new BEST OF BEEBUG disc is based on the recent series of articles and programs dealing with ASTAAD, our comprehensive CAD program for the Master and Compact. The ASTAAD disc contains everything you need to use this very impressive piece of software, including the enhanced versions of the ASTAAD and STEAMS programs. This is supplied complete with documentation and printed keystrips for use with ASTAAD and STEAMS.



```
DEL  Text Line Edge Outl S Fore Fix  Accl Abs  x=  1003.0 y=   446.t
File name? TDOHE3          Scale:  1.00times    Vector:   363.8
```

ASTAAD £9.95

* Enhanced ASTAAD CAD program now including full mouse and joystick control, built-in printer dump and speed improvement.
* STEAMS image manipulator
* 8 page manual covering the use of the system.
* Keystrips for ASTAAD and STEAMS
* Sample picture files to experiment with.



```
Save  MOVE Text copy Line Edge Outl S Fore Fix  Accl Rel  x=  391.6 y=  -52.8
Filename DIFFAMP5              Scale:100.00units/inDiagonal   475.1
```

```
PL1        R.S. 475-268 (1)
SK1        R.S. 475-303 (1)
PL2        R.S. 475-280 (1)
PINS       R.S. 466-797 (2)
SKTS       R.S. 466-826 (2)
SHELLS     R.S. 475-347 (2)
IC1        LF 412CN     (1)
IC2        NE 531CN     (1)
CAPACITORS R.S. 125-733 (4)
RESISTORS  0.25W METAL FILM 1% TOLERANCE
```

```
Preliminary circuit design  16/1/89
DIFFERENTIAL AMPLIFIER
```

Recent magazines have shown many examples of the truly impressive results that can be achieved with ASTAAD for producing technical drawings and diagrams including perspective work. The disc also contains several example drawings for you to load and modify, for easy familiarisation with the software. All this is at a fraction of the price that you would need to pay for equivalent commercial software.

Our first two releases under the Best of BEEBUG label are still available:

General Utilities £5.75
Best of BEEBUG Disc 1

* Printer Buffer * * ROM Controller * Sprite Editor/Animator * Multi-Character Printer Driver for View * Mode 7 Screen Editor * Multi-Column Printing * Epson Character Definer * BEEBUG MiniWimp † * ROM Filing System Generator

*** Master series only. † Requires sideways RAM.**

Applications £5.75
Best of BEEBUG Disc 2

* Business Graphics * Video Cataloguer * World by Night and Day * Phone Book * Page Designer * Personalised Letter-Heads * Mapping the British Isles * Selective Breeding * Appointments Diary * The Earth from Space * Personalised Address Book

Please rush me my Best of BEEBUG discs:

ASTAAD Code 1407A *(80 track DFS)* ☐
Code 1408A *(3.5" ADFS)* ☐

General Utilities Disc Code 1405A ☐
Applications Disc Code 1404A ☐

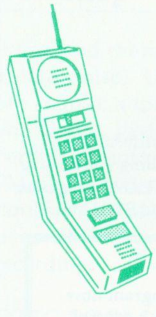
Name ____________________
Address ____________________
____
Memership No ____________________

Total £________
Postage (£0.60 + £0.30 for every other disc) £________
Grand Total £________

I enclose a cheque for £__________ OR please debit my Access, Visa or Connect account, Card No_______/_______/_______/_______ Expiry_____/____ Signed__________________

Return to BEEBUG Ltd, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX. Telephone (0727) 40303.

The Comms Spot



by Peter Rochford

This month's Comms Spot is a pot pourri of news and general information, rather than concentrating on any particular topic. We kick off with the latest on Prestel and Micronet.

Prestel has announced that, as from next month, all its access ports will feature full Vasscom error correction for those who have the software to utilise it. Unfortunately I know of no software for the Beeb, but BEEBUG's Hearsay comms package for the Archimedes certainly has this feature. Nevertheless, this does mean that all the ports will allow connection at baud rates of 1200/75, 1200/1200 and 2400/2400. Perhaps the price of high speed modems will now fall even further. Good news indeed.

Moving on to Micronet. There is a new editor of the BBC Microbase area, and there are now far more regular updates with the promise of new features. There have already been moans however, that too much of the BBC Microbase area is now being devoted to the Archimedes. Of course, the fact is that the Arc is a relatively new machine which is rapidly growing in popularity, and there are plenty of new products to feature, while few new hardware or software products are appearing for the good old Beeb.

Our own BEEBUG area on Micronet has now finally undergone its complete facelift and updates are now more regular, I am pleased to say. The retail area has been divided up so that Arc and Beeb products are available in separate sections. Also, there are more choices from the retail menus which save wading through pages of products to get to the one that you want. By the way, it can be far quicker and in some instances cheaper to order your goods via the response frame on Micronet than by post or by phone.

New telesoftware has been lacking for some time on our area. This has been due to quite a few factors, but mainly to Micronet's decision to stop supporting chargeable telesoftware, as they had found this to be uneconomic. The whole BEEBUG area was financed largely by revenue from chargeable telesoftware. Now that we cannot charge for telesoftware we have had to make a decision on whether to keep the area going. We have decided for it, and we do intend to provide more telesoftware for free downloading. This will be pursued when Micronet sorts out the technical problems with its new telesoftware gateway.

This new gateway differs from the old method of providing telesoftware on Micronet by using an external computer (an Amiga!) which you access through Micronet. This has not been entirely successful, and I have yet to get connected to it. Instead, having gone through the gateway I have been left with a blank screen and flashing cursor, and unable to go back to Prestel except by executing a Ctrl-Break and logging on again. Not good.

Still on the subject of Prestel and Micronet, those who are subscribers will be only too well aware of the recent introduction of time charges and the earlier large increases in the subscription rates. This has led to a general hue and cry with many people stopping their subscriptions altogether. Now with the recent budget, the Chancellor has made electronic information and Email service charges subject to VAT. In consequence, the cost of going on-line, particularly with BT, has risen sharply in the last year or so. This latest increase will no doubt prompt more people into questioning whether subscribing to Prestel and Micronet is really worth the money, and may lead to further reduction in numbers of users. Rather sad and depressing.

One thing we have never featured in our Comms Spot is the France Telecom Minitel service, although I am sure many of you have heard or know about it. For those that don't, it is the French equivalent of Prestel. France Telecom has however, provided some four million terminals totally free to encourage and promote its use by the French public. It has been very successful indeed in the few years it has been operating, and now far outstrips Prestel with its 97,000 regular subscribers.

Minitel provides a variety of services including Email, teleshopping and access to a database of telephone subscribers enabling searching by name or address etc. Recently Minitel was the subject of a French government investigation, when accusations were made that it was being used by call girls to obtain clients. Maybe that was why it has been so successful!

Whatever the reason for its success, it now appears that BT has been closely monitoring the progress of Minitel and plans to launch its own equivalent service in the UK. This apparently would use the same kind of technology as Prestel, which it would no doubt totally replace. It is rumoured that unlike Minitel, BT would charge for terminals (typical BT!), but this charge would only be 'nominal'.

To help with the investment in such a venture, it appears that BT are looking into the possibility of taking on partners, such as banks and/or mail order companies. Home banking and teleshopping have enjoyed a reasonable degree of success in the UK. My own personal feeling is that this is the way to go, and would be successful, provided the terminals were low cost and easy-to-use with no hefty time charges. Also, comms is something the vast majority of the public know little or nothing about. The idea of plugging anything other than a telephone or an answering machine into the BT socket is a pretty alien idea to most. It will need some careful marketing and advertising to convince Joe Public that this is a 'good thing', and that they are not being conned into some new way-out technology that they don't really need and have little chance of understanding.

If BT go ahead, it would be an exciting development and could lead to a massive upsurge in the use of comms in this country - something I would personally welcome. It would be nice though if BT were to abandon the existing viewdata standard and adopt the Minitel videotex, as have done in Germany and Spain. We could then look forward to the possibility of our system being linked to other European ones.

Some time back in a previous Comms Spot I examined the subject of modems and modem standards. I mentioned that Amstrad had released a low cost V22bis card for PCs and hinted that perhaps they would eventually release a standalone version of it. Well they have, and it is priced at £249 plus VAT. I am in the process of getting hold of one for review and that should appear very shortly.

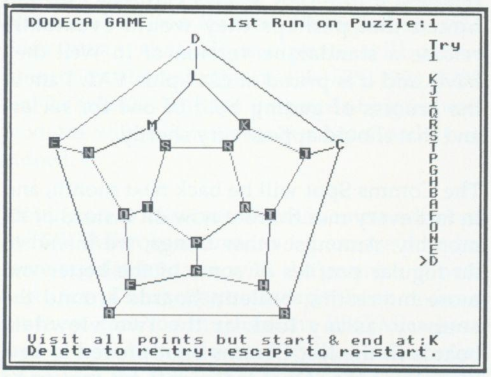
The Comms Spot will be back next month, and in fact every month from now on instead of bi-monthly. Amongst other things, we intend to do regular profiles of some of the better and more interesting bulletin boards around the country, as my look at the two viewdata boards in the last Comms Spot proved to be a popular idea. One we won't be featuring however, is called the Erogenous Zone which has now been brought to the attention of MP's because of the 'pornographic' nature of its content. It is hardly surprising I suppose that sooner or later someone would come up with this sort of thing. What will we have next? Digitised pictures of Page 3 girls on Prestel? Might improve the subscription figures though!

If you have any comments about the Comms Spot column you wish to make, please feel free to write to me c/o BEEBUG or mailbox me on Prestel (019996601). I am always interested in any ideas you may have and also any comms news that I can include in the column. See you next month!

The Dodeca Game

Jim Proctor presents an unusual challenge to your powers of thought and concentration.

An early version of this puzzle was developed as the Icosian Game by the mathematician Sir W.R.Hamilton in 1859. It was played using a regular solid dodecahedron with place names at each of the solid's 20 corners. Starting at any corner, the object of the game was to make a 'trip round the world'- visiting each vertex once and only once, before returning to the starting point. The program listed here provides a computerised version of the same game, with some additional twists not possible with the original.

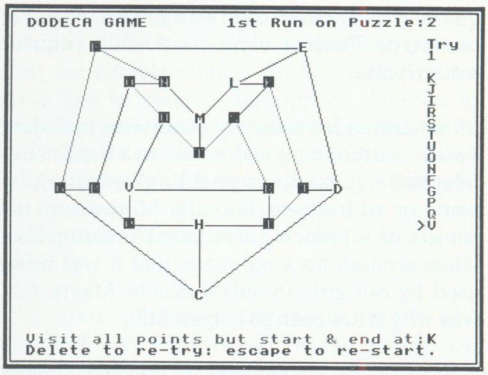


```
DODECA GAME        1st Run on Puzzle:1
                                   Try
                                   1

                                   K
                                   J
                                   I
                                   Q
                                   R
                                   S
                                   T
                                   P
                                   G
                                   H
                                   B
                                   A
                                   F
                                   O
                                   N
                                   E
                                  >D

Visit all points but start & end at:K
Delete to re-try: escape to re-start.
```

PLAYING THE DODECA GAME

The program should be straightforward to type in, but do be careful with the data statements. In this computerised version, the dodecahedron is first set out as a flat network that is topologically identical with the edges of the regular solid figure. But, as you will see, later versions are decidedly irregular! A complete game consists of 3 runs on each of 3 different dodecahedral puzzles. Each route must be different, and reverse order routes are not allowed. A new game starts afresh with a new combination of puzzles to solve.

You can abandon any game by pressing Escape, while Shift-Escape will exit from the program altogether. That's all there is to it, but you will need to concentrate quite hard, particularly with the topological distortions to find a solution quickly.



```
DODECA GAME        1st Run on Puzzle:2
                                   Try
                                   1

                                   K
                                   J
                                   I
                                   R
                                   S
                                   T
                                   U
                                   O
                                   N
                                   F
                                   G
                                   P
                                   Q
                                  >V

Visit all points but start & end at:K
Delete to re-try: escape to re-start.
```

NOTE: The program needs to run from PAGE=&E00. Where necessary, the program will automatically relocate itself in memory when first run. The magazine disc/tape also contains an additional program which displays information about the game on-screen.

```
   10 REM Program DODECA
   20 REM Version B 1.1
   30 REM Author Jim Proctor
   40 REM BEEBUG April 1988
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 320
  110 IFPAGE>&E00 PROCpage:END
  120 *FX3,0
  130 MODE1
  140 PROCinit
  150 start=TRUE
  160 REPEAT:PROCrerun
  170 REPEAT:start=FALSE
  180 REPEAT:PROCsetup
  190 REPEAT:PROCreset
  200 PROCdraw:PROClabel
  210 REPEAT
  220 REPEAT:PROCkey
  230 UNTIL key=TRUE:C%=C%+1
  240 UNTIL C%=20:PROCcheck
  250 UNTIL reset=FALSE
  260 PROCdone:pzl%=pzl%+1
  270 UNTIL pzl%>3
  280 pzl%=1:run%=run%+1
  290 UNTIL run%>3:PROCendgame
  300 UNTIL FALSE
  310 :
  320 IF ERR=17 AND INKEY-1 MODE3:END
  330 IF ERR=17 GOTO150
  340 MODE7:REPORT:PRINT" at line ";ERL
  350 END
  360 :
 1000 DEF PROCinit
 1010 DIMx%(20),y%(20),p%(20,3)
 1020 DIMtime%(3,3),try%(3,3),P$(3,3)
 1030 run%=1:pzl%=1:best%=60
 1040 VDU5,19,3,2;0;19,1,5;0;
 1050 VDU23,255,255,255,255,255,255,255,
255,255
 1060 FOR i=1 TO 20
 1070 FOR j=1 TO 3
 1080 READp%(i,j)
 1090 NEXT
 1100 NEXT
 1110 ENDPROC
 1120 DATA2,5,6,1,3,8,2,10,4,3,12,5,4,14
,1,1,15,7,6,16,8,7,2,9,8,17,10,9,3,11,10
,18,12,11,4,13,12,19,14,13,5,15,14,20,6,
20,7,17,16,9,18,17,11,19,18,13,20,19,15,
16
 1130 :
 1140 DEF PROCsetup
 1150 CLS
 1160 V%=62+2*pzl%
 1170 C%=1:try%(run%,pzl%)=0:f%=100
 1180 IF pzl%=1 RESTORE1250:S%=5+R3%:f%=
10:ELSE IF pzl%=2 PROCpuz2:ELSE PROCpuz3
 1190 FOR N=1 TO 20
 1200 READ x%,y%
 1210 x%(N)=f%*x%:y%(N)=f%*y%
 1220 NEXT
 1230 L$=CHR$(V%+S%):P$=""
 1240 ENDPROC
 1250 DATA24,15,74,15,90,63,49,92,8,63,3
0,23,49,27,68,23,70,43,80,60,63,68,49,82
,36,68,18,60,28,43,49,35,63,45,58,62,40,
62,35,45
 1260 DATA1,2,9,2,9,5,9,9,1,7,2,3,4,4,8,
3,8,5,8,6,6,6,4,7,3,7,2,6,3,5,6,5,7,4,7,
5,4,6,4,5
 1270 DATA1,2,9,2,9,9,5,7,1,9,4,3,8,3,8,
6,8,7,7,6,4,6,3,7,2,8,2,7,2,4,7,4,7,5,4,
5,3,5,3,4
 1280 DATA1,2,9,2,9,8,5,9,1,9,2,3,5,3,8,
3,7,6,8,7,6,8,5,8,2,8,2,6,2,4,5,5,7,4,6,
7,4,7,3,4
 1290 DATA5,2,9,5,8,9,2,9,1,5,5,4,7,4,8,
5,7,8,6,8,5,7,4,8,3,8,2,5,3,4,7,5,6,7,5,
6,4,7,3,5
 1300 DATA1,2,8,2,9,8,1,9,3,6,4,3,7,4,8,
6,8,7,7,8,5,8,3,8,4,7,3,5,3,3,5,4,6,5,6,
7,5,6,4,5
 1310 DATA1,2,9,2,7,5,9,9,1,9,2,5,2,3,6,
3,5,5,6,6,7,8,6,8,4,8,2,8,2,6,3,4,4,5,5,
7,4,7,3,7
 1320 DATA1,2,5,3,9,2,9,8,1,9,3,3,4,5,6,
5,7,3,8,4,8,7,3,8,2,7,2,4,2,3,4,6,7,6,6,
7,3,6,3,4
 1330 DATA1,2,9,2,9,7,5,9,1,7,2,3,5,3,8,
3,8,5,8,6,6,7,5,7,4,7,2,6,2,5,5,4,7,5,6,
6,4,6,3,5
 1340 :
 1350 DEF PROCpuz2
 1360 S%=R3%
 1370 IF R1%=1 RESTORE1260:ELSE IF R1%=2
 RESTORE1270:ELSE IF R1%=3 RESTORE1280:E
LSE RESTORE1290
 1380 ENDPROC
 1390 :
 1400 DEF PROCpuz3
 1410 S%=10+R3%
 1420 IF R2%=1 RESTORE1300:ELSE IF R2%=2
 RESTORE1310:ELSE IF R2%=3 RESTORE1320:E
LSE RESTORE1330
 1430 ENDPROC
 1440 :
 1450 DEF PROCdraw
 1460 PROCline(3,5,1,5)
 1470 PROCline(3,15,6,15)
 1480 PROCline(3,20,16,20)
 1490 RESTORE1550
 1500 FOR N=1 TO 10
 1510 READi,j
 1520 PROCline(3,i,j,j)
 1530 NEXT
 1540 ENDPROC
 1550 DATA1,6,2,8,3,10,4,12,5,14,16,7,17
9,18,11,19,13,20,15
 1560 :
 1570 DEF PROCline(col%,sp%,ip%,fp%)
 1580 GCOL0,col%
 1590 MOVEx%(sp%),y%(sp%)
 1600 FOR n=ip% TO fp%
 1610 DRAWx%(n),y%(n)
 1620 NEXT
 1630 ENDPROC
 1640 :
 1650 DEF PROClabel
 1660 IF run%=1 R$="st":ELSE IF run%=2 R
$="nd":ELSE R$="rd"
 1670 VDU4,23,1,0;0;0;0;
 1680 COLOUR2
 1690 PRINTTAB(0,1)"DODECA GAME"SPC7;run
%;R$" Run on Puzzle:";pzl%;TAB(36,3)"Try
";
 1700 COLOUR1
 1710 PRINTTAB(36,4);try%(run%,pzl%)+1TA
B(0,29)"Visit all points but start & end
 at:";:VDU17,2,V%+S%
 1720 COLOUR3
 1730 PRINTTAB(0,30)"Delete to re-try: e
scape to re-start."
 1740 VDU5
 1750 FOR N=1 TO 20
 1760 PROCletter(N,20,0,16,2)
 1770 NEXT
 1780 SOUND1,-15,150,8
 1790 REPEAT
 1800 G%=GET
 1810 IF G%<>V%+S% SOUND1,-15,0,4
 1820 UNTIL G%=V%+S%
 1830 PROCletter(S%,20,1,16,2)
 1840 VDU4,23,1,0;0;0;0;
 1850 COLOUR1
 1860 PRINTTAB(36,6);:VDUV%+S%,5
 1870 SOUND1,-15,200,2
 1880 TIME=0:G%=S%:C%=1
 1890 ENDPROC
 1900 :
 1910 DEF PROCreset
 1920 VDU4,23,1,0;0;0;0;
 1930 IF P$<>"" PROCseetry
 1940 IF P$(1,pzl%)<>""OR P$(2,pzl%)<>""
 PROCseerun
 1950 VDU5:ENDPROC
 1960 :
 1970 DEF PROCletter(Z%,d1%,c1%,d2%,c2%)
 1980 MOVEx%(Z%)-16,y%(Z%)+d1%
 1990 VDU18,0,c1%,255
 2000 MOVEx%(Z%)-16,y%(Z%)+d2%
 2010 VDU18,0,c2%,V%+Z%
 2020 ENDPROC
 2030 :
 2040 DEF PROCkey
 2050 key=TRUE:reset=FALSE
 2060 K%=GET-V%
 2070 IF K%=127-V% C%=19:reset=TRUE:VDU7
:ENDPROC
 2080 IF p%(G%,1)<>K% ANDp%(G%,2)<>K% AN
Dp%(G%,3)<>K% key=FALSE:SOUND1,-15,0,4:E
NDPROC
 2090 IF C%<20ANDPOINT(x%(K%)-12,y%(K%)+
12)=1ORPOINT(x%(K%),y%(K%)+12)=1 key=FAL
SE:SOUND1,-15,0,4:ENDPROC
 2100 GCOL0,1
 2110 MOVEx%(G%),y%(G%)
 2120 DRAWx%(K%),y%(K%)
 2130 PROCletter(G%,20,2,16,1)
 2140 G%=K%:PROCletter(G%,16,0,20,1)
 2150 SOUND1,-15,200,2
 2160 VDU4:COLOUR2
 2170 PRINTTAB(35,5+C%)" "TAB(35,6+C%)">
"
 2180 COLOUR1
 2190 PRINTTAB(36,6+C%);:VDUV%+K%,5
 2200 L$=L$+CHR$(V%+K%)
 2210 ENDPROC
 2220 :
 2230 DEF PROCretry
 2240 reset=TRUE:VDU4
 2250 IF go=FALSE PRINTTAB(0,29)SPC79TAB
(2,29)"That's NOT available - TRY AGAIN!
";:ELSEPRINTTAB(0,29)SPC79TAB(2,29)"You
didn't finish at "CHR$(V%+S%)" - TRY AGA
IN!";
 2260 FOR N=100 TO 0STEP-1
 2270 SOUND1,-N/10,N,1
 2280 NEXT
 2290 ENDPROC
 2300 :
 2310 DEF PROCdone
 2320 P$(run%,pzl%)=P$
 2330 PROCline(1,K%,S%,S%)
 2340 PROCletter(K%,20,2,16,1)
 2350 VDU4:COLOUR1
 2360 PRINTTAB(35,25)" "TAB(36,26);:VDUV
%+S%
 2370 time%(run%,pzl%)=TIME/100
 2380 COLOUR2
 2390 PRINTTAB(31,26)"OK ->"TAB(31,28)"T
ime:";time%(run%,pzl%);
 2400 COLOUR3
 2410 PRINTTAB(0,29)SPC79TAB(1,29)"Press
SPACE to continue...";
 2420 FOR N=0 TO 150STEP4
 2430 SOUND1,-N/10,N,1
 2440 NEXT
 2450 REPEAT UNTIL INKEY-99
 2460 ENDPROC
 2470 :
 2480 DEF PROCendgame
 2490 CLS:COLOUR2
 2500 PRINTTAB(6,2)"T H E  D O D E C A
G A M E"
 2510 FOR run%=1 TO 3
 2520 IF run%=1 R$="st":ELSE IF run%=2 R
$="nd":ELSE R$="rd"
 2530 COLOUR1
 2540 PRINTTAB(0,7*run%-3);run%;R$;" RUN
 on..."
 2550 FOR pzl%=1 TO 3:COLOUR1
 2560 PRINTTAB(0,7*run%-1)"Puzzle Time+E
xtras ";
 2570 COLOUR3
 2580 PRINT"Solutions for Run: ";run%
 2590 P$="":FOR N=1 TO 20
 2600 P$=P$+CHR$(32+ASC(MID$(P$(run%,pzl
%),N,1)))
 2610 NEXT
 2620 COLOUR2
 2630 PRINTTAB(0,7*run%-1+pzl%)"No. ";pz
l%;TAB(9);time%(run%,pzl%);TAB(14);10*tr
y%(run%,pzl%);
 2640 PRINTTAB(19)P$
 2650 NEXT
 2660 NEXT
 2670 tot%=0
 2680 FOR run%=1 TO 3
 2690 FOR pzl%=1 TO 3
 2700 tot%=tot%+time%(run%,pzl%)+10*try%
(run%,pzl%)
 2710 NEXT
 2720 NEXT
 2730 avtime%=tot%/9
 2740 IF best%>avtime% best%=avtime%
 2750 COLOUR2
 2760 PRINTTAB(0,28)"Average Time:";avti
me%;"secs"TAB(21,28)"Best Time:";best%;"
secs"
 2770 COLOUR1
 2780 PRINTTAB(0,30)"Press return to set
 up a fresh game...";
 2790 run%=1:pzl%=1:REPEAT UNTIL INKEY-7
4
 2800 ENDPROC
 2810 :
 2820 DEF PROCcheck
 2830 P$=L$:L$=CHR$(V%+S%):go=TRUE
 2840 RP$=LEFT$(P$,1)
 2850 FOR N=20 TO 2STEP-1
 2860 RP$=RP$+MID$(P$,N,1)
 2870 NEXT
 2880 IF P$=P$(1,pzl%)OR RP$=P$(1,pzl%)
go=FALSE:PROCretry:ELSE IF p%(S%,1)=K% O
Rp%(S%,2)=K% ORp%(S%,3)=K% reset=FALSE:E
NDPROC
 2890 IF K%<>127-V% AND go=TRUE PROCretr
y
 2900 try%(run%,pzl%)=try%(run%,pzl%)+1
 2910 ENDPROC
 2920 :
 2930 DEF PROCseetry
 2940 COLOUR3
 2950 FOR Y%=6 TO 25
 2960 PRINTTAB(35,Y%)SPC4TAB(38,Y%)MID$(
P$,Y%-5,1)
 2970 NEXT
 2980 ENDPROC
 2990 :
 3000 DEF PROCseerun
 3010 COLOUR2
 3020 PRINTTAB(32,3)"Run";
 3030 COLOUR3
 3040 PRINTTAB(32,4);1;:IF run%=3 PRINTT
AB(34,4);2;
 3050 FOR r%=1 TO 2
 3060 FOR Y=6 TO 25
 3070 PRINTTAB(30+2*r%,Y)MID$(P$(r%,pzl%
),Y-5,1)
 3080 NEXT
 3090 NEXT
 3100 ENDPROC
 3110 :
 3120 DEF PROCrerun
 3130 R1%=RND(4):R2%=RND(4):R3%=RND(10)
 3140 FOR i=1 TO 3
 3150 FOR j=1 TO 3
 3160 P$(i,j)=""
 3170 NEXT:NEXT
 3180 IF start=TRUE ENDPROC
 3190 IF avtime%<best% best%=avtime%
 3200 ENDPROC
 3210 :
 3220 DEF PROCpage
 3230 *FX3,2
 3240 *KEY0 *T.|MF.A%=0TO(TOP-PA.)S.4:A%
!&E00=A%!PA.:N.|MPA.=&E00|MO.|MRUN|M
 3250 *FX138,0,128
 3260 ENDPROC
 1380 PRINT'"2. LR CIRCUIT"
 1390 INPUT''"Selection ",S
 1400 IF S<1 OR S>2 GOTO1350
 1410 PRINTTAB(0,18)CHR$133"* Illegal En
tries *"'CHR$134"Negative or zero"'CHR$1
34"quantities."
 1420 PRINTTAB(0,12)SPC39:PRINTTAB(0,12)
CHR$131" Frequency (in hertz) ";:INPUT F
R
 1430 IF FR<=0 VDU7:GOTO1420
 1440 PRINTTAB(0,13)SPC39:PRINTTAB(0,13)
CHR$134" R.M.S Voltage (in volts) ";
 1450 INPUT V:VMAX=V*SQR2
 1460 IF V<=0 VDU7:GOTO1440
 1470 PRINTTAB(0,14)SPC39:PRINTTAB(0,14)
CHR$130" Resistance (in ohms) ";:INPUT R
E
 1480 IF RE<=0 VDU7:GOTO1470
 1490 IF S=1 PROCcapres ELSE PROCindres
 1500 IF VR<VC THEN1550
 1510 S$="RESISTOR":s=2
 1520 T$=T1$:t=1
 1530 VR=250:VC=VC*250/vr
 1540 ENDPROC
 1550 S$=T1$:s=1
 1560 T$="RESISTOR":t=2
 1570 VC=250:VR=VR*250/vc
 1580 ENDPROC
 1590 DEFPROCcapres
 1600 PRINTTAB(0,15)SPC39:PRINTTAB(0,15)
CHR$129" Capacitance (in micro-farads) "
;
 1610 INPUT CA:CA=CA/1E6
 1620 IF CA<=0 VDU7:GOTO1600
 1630 REAC=1/(2*PI*FR*CA)
 1640 PROCcalc
 1650 PD=-90:T1$="CAPACITOR"
 1660 PHASE=-DEGATN(REAC/RE)
 1670 PH$="lags behind"
 1680 ENDPROC
 1690 DEFPROCindres
 1700 PRINTTAB(0,15)SPC39:PRINTTAB(0,15)
CHR$129" Inductance (in milli-Henries) "
;
 1710 INPUT L:L=L/1000
 1720 IF L<=0 VDU7:GOTO1700
 1730 REAC=2*PI*FR*L
 1740 PROCcalc
 1750 PD=90:T1$="INDUCTOR "
 1760 PHASE=DEGATN(REAC/RE)
 1770 PH$="leads"
 1780 ENDPROC
 1790 DEFPROCpower(P,C,H)
 1800 GCOL3,C
 1810 FORZ%=0TO720STEP4
 1820 A=H*SINRAD(Z%+P)*SINRADZ%
 1830 MOVE275+Z%,0
 1840 DRAW275+Z%,A
 1850 NEXT
 1860 ENDPROC
 1870 DEFPROCcalc
 1880 Z=SQR(RE^2+REAC^2)
 1890 IMAX=VMAX/Z
 1900 VR=IMAX*RE:vr=VR
 1910 VC=IMAX*REAC:vc=VC
 1920 ENDPROC
 1930 DEFPROCinf
 1940 PRINTTAB(2)'"The impedance triangl
e for a"TAB(2)"RESISTOR/"T1$" series cir
cuit"
 1950 GCOL0,2
 1960 VDU5
 1970 MOVE250,512:DRAW500,512
 1980 PLOT0,32,16
 1990 PRINT"I  or V"CHR$10;"r"
 2000 MOVE250,512
 2010 DRAW250,512+350*SINRAD(PHASE)
 2020 PLOT0,-16,32*SINRAD(PHASE)
 2030 PRINT"V"CHR$10"x"CHR$11"=IX"
 2040 GCOL0,3:MOVE250,512
 2050 DRAW500,512+350*SINRAD(PHASE)
 2060 PLOT0,16,0:PRINT"V=IZ"
 2070 MOVE300,496-62*(PD=90)
 2080 VDU225,4,23,1;0;0;0
 2090 PROCready(25,23)
 2100 ENDPROC
 2110 DEFPROCprint
 2120 @%=&020200
 2130 PRINTTAB(0,1)"Reactance X = ";REAC
;:VDU32,226
 2140 PRINT''"Impedance Z = ";Z;:VDU32,2
26
 2150 PRINT'''TAB(4)"R.M.S Voltage = ";V
" Volts"
 2160 PRINT'"R.M.S Current = ";IMAX*1000
/SQR2;" mA"
 2170 PRINT''"Power Phase Angle "CHR$225
"=";PHASE;" degrees"
 2180 PRINT"Giving a power factor of ";C
OS RAD PHASE
 2190 PRINT'"a.c Frequency ";FR;" Hz"
 2200 PRINTTAB(17,20)"voltage=";vc" V"TA
B(17,27)"voltage=";vr" V"
 2210 COLOUR2:PRINTTAB(0,5)"In the RESIS
TOR/"T1$" series circuitthe"
 2220 PRINT"and ";PH$;" the"''''"by the"
 2230 VDU28,17,30,38,19
 2240 PRINTT1$''''"and "PH$'''"RESISTOR"
''"by ";PD" degs"
 2250 VDU26
 2260 PROCready(29,30)
 2270 ENDPROC
```

RISC USER

The Archimedes Support Group

Our Risc User magazine is now in its second volume and is enjoying the largest circulation of any magazine devoted to the Archimedes. The list of members seeking support from the Risc User group is growing steadily and as well as private individuals includes schools, colleges, universities and industry and government establishments.

Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines. A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer, particularly at this time while documentation on the Archimedes and RISC OS is still limited.

Here are just some of the topics covered in more recent issues of RISC User:

A MULTI-TASKING NOTE-PAD

RAY TRACING
The program that produces the famous molecule demo.

THE R140 UNIX WORKSTATION
A look at Acorn's entry into the business market.

NEW FONT STYLES MODULE

RISC OS
A selection of articles covering all aspects of the new Archimedes opeating system, including writing multi-tasking programs.

MOVIE MAKER
A complete animation system which is a culmination of a series on graphics and delta file techniques.

RISC USER TOOLBOX
A comprehensive toolbox module for the Archimedes.

MAKING THE MOST OF LOGISTIX
How to use one of the best spreadsheets for the Archimedes.

MOUSE MENU
A general purpose WIMP based menu, which is fully RISC OS compatible.

ARCHIMEDES VISUALS
A series of articles with some stunning visual effects.

ARCADE
A survey of the games software scene for the Archie.

ART NOUVEAU
A look at this impressive art package from CAL.

USER SPRITES
More flexible use of the sprite system.

RISC FORTH REVIEW
An investigation into this alternative language.

Don't delay - Phone your instructions now on (0727) 40303

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.50 (overseas see below).

Name:...
Memb No:...
Address: ....
....
....
....

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.50
Rest of Europe and Eire	£13.00
Middle East	£15.00
Americas and Africa	£17.00
Elsewhere	£19.00

I wish to receive both BEEBUG and RISC User. *I enclose a cheque for £ .... or alternatively*

I authorise you to debit myACCESS/Visa/Connect account: ______/ ______/ ______/ ______/

Signed: .... *Expiry Date:* _____/ _____/

***Send to:** RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX, or telephone (0727) 40303* options, the cheapest costing around £5000. The top of the range model even includes its own hard disc drive to store a wide range of fonts. Incidentally, all the pages of this magazine are printed at A4 size on an Apple Laser Writer and then reduced down by the printers.

A final word on cost is shop around - there are many good deals on laser printers to be found.

CONCLUSION

Do you need a laser printer? The answer is it depends on what you want to do. Many people find that a simple nine-pin dot matrix printer serves them perfectly well. Others, particularly those who use their systems mainly for letter writing, will get by quite happily with a 24-pin dot matrix unit, or a daisywheel printer. There are however several classes of user that could justify the purchase of a laser printer. Firstly, used as a text only printer, a laser printer is many times faster than any other type. In some cases speeds of up to twelve A4 pages a minute can be achieved. This, coupled with the quiet operation, makes the laser printer perfect for use as an office printer.

The second class of prospective users are those who want to produce graphical output using existing software, but to a better standard than they can achieve with a normal printer (300 dots per inch instead of about 70).

The third category are people who need the page description facilities of PostScript, normally in order to use Desktop Publishing systems. However, the limited memory of the Beeb makes this impractical, and while it is highly likely that software containing PostScript drivers will appear for the Archimedes, there will be none for the model B.

B

Workshop - Spin a Disc (continued from page 50)

When a disc is formatted, this is done track by track. The formatting of individual tracks is then synchronised to the index hole pulse from the drive. The controller waits for the index pulse to occur, writes a certain number of bytes to form a *Post index gap*, and then writes each sector in turn, leaving gaps between each. When all the sectors have been written (10 for a DFS disc, 16 for ADFS), the controller writes dummy bytes until the index pulse occurs again, at which point the complete track has been formatted. The reason for the gap between sectors is so that when a data field has been written, the drive can be switched back to read mode before accidentally erasing the ID field of the next sector.

SKEW ADJUSTMENT

When formatting a disc, there is a trick the formatter can play to speed up subsequent reading and writing of the disc. Firstly, consider reading a file that consists of several sectors, and which spans more than one track. When the last sector of one track is reached, the head must be moved to the next track and the first sector of that read. It takes several milliseconds to step from one track to the next, and by this time the first sector of the next track may have already passed the drive head. The controller then has to wait for another revolution of the disc before the desired sector is found.

The solution to this is to skew the positioning of sectors on each track. For example, suppose we had ten sectors per track, and the first sector after the index hole on track zero was sector zero. We could then make the first sector on track one sector seven, the first on track two sector four and so on. Now, when you get to the end of track zero, you have the time taken for three sectors to pass under the head before sector zero is reached. This should give ample time for stepping the head.

B

This month's hints and tips have been collected together by Mike Williams. Remember that we pay five pounds for each hint published and fifteen pounds for the star hint.

***** STAR HINT *****

CREATING CONDITIONAL BOOT FILES

Philip Hetherington

The following example (for a Master or Compact using the ADFS) of coding for a !BOOT file shows how it can be set up to load specified ROM images into sideways RAM the first time that it is called but skips this on subsequent calls. As a test it uses the fact that an active ROM has a non-zero value in the corresponding page two address (between &2A1 and &2B0). Thus for ROM slot 0 test &2A1, for ROM slot 1 test &2A2, and so on.

```
*BASIC
J%=?&2A5:K%=?&2A6:L%=?&2A7:M%=?&2A8
IF J%=0 THEN *ADFS
IF J%=0 THEN *DIR :0.$
IF J%=0 THEN *LIB LIBRARY
IF J%=0 THEN *SRLOAD $.ROMS.rom1 8000 4 Q
IF K%=0 THEN *SRLOAD $.ROMS.rom2 8000 5 Q
IF L%=0 THEN *SRLOAD $.ROMS.rom3 8000 6 Q
IF M%=0 THEN *SRLOAD $.ROMS.rom4 8000 7 Q
IF M%=0 THEN VDU2,1,27,1,67,1,66,3
IF J%=0 OR K%=0 OR L%=0 OR M%=0 THEN
  CH."ADFSMEN" ELSE *CAT
```

This assumes that the ROM images are held in an ADFS directory called ROMS, and are loaded into sideways RAM slots 4, 5, 6 and 7. Substitute the names of your ROM images for rom1, rom2 etc. As a further example, after the ROM images have been loaded, the BOOT file conditionally sets the page length on the printer to 66 lines.

So Shift-Break to load your ROM images and perform other once only tasks before chaining a menu program. Then press Ctrl-Break to initialise the ROMs, and subsequent boots will perform only the *CAT (or any alternative you include).

SPELLMASTER/QUICKCALC CLASH

V.F.Lane

Having SpellMaster installed prevents Quickcalc from working correctly. The solution is to use: *WORKOFF followed by Break, and then Shift-Break as normal to boot Quickcalc. On leaving Quickcalc, Break followed by *WORKON and Break again will restore SpellMaster to normal.

TESTING MACHINE TYPE

If you want to determine on which machine your program is running, this can be done using INKEY-256 (see First Course in Vol.7 No.9). The complete list of values returned and their meaning is as follows:

Dec.	Hex	Meaning
0	&00	BBC micro MOS 0.10
1	&01	Acorn Electron MOS 1.00
-1	&FF	BBC micro MOS 1.00 or 1.20
254	&FE	BBC micro MOS 1.00 or 1.10 USA
253	&FD	BBC Master 128 MOS 3.20
252	&FC	BBC micro MOS 1.20 W.Germany
251	&FB	BBC micro B+ MOS 2.00
250	&FA	Acorn ABC MOS
245	&F5	BBC Master Compact MOS 5.10
244	&F4	Master 128 MOS 3.26
160	&A0	Archimedes Arthur 1.20
161	&A1	Archimedes RISC OS

As an example of its use:

```
        IF INKEY-256=245 THEN PRINT"Compact"
```

Note that the abbreviation MOS stands for Machine Operating System.

MULTIPLE COPIES WITH WORDWISE PLUS

Al Harwood

Using Wordwise Plus to produce multiple copies of a piece of text by selecting menu option 6 each time can be laborious. The following short segment program will produce as many copies as you require.

```
REM Multiple copy printer
P.
P.
P."Enter number of copies"
A$=GLK$
S.TE.
D.
PRE.TE.
P."Press any key to continue"
A$=GET
T.VALA$
```

Select a segment, enter this program and save for future reference. With the text to be copied and this segment program loaded press the corresponding function key to activate the segment program.

Dabhand User News

Now Available! • Master 512 User Guide • Shareware Volume 2
Mastering Interpreters & Compilers • Mini Office II Guide • BASIC V Guide

Master 512: User Guide

At long last, a comprehensive guide for all users of the Master 512. Highly practical in approach, this book provides detailed information on all DOS Plus commands, and explains how they differ from MS-DOS. It shows step-by-step how to install and run PC applications on the Master 512, including useful techniques such as the creation of batch files.

In addition, the use and operation of the utilities provided with the machine are explained, many of which are previously undocumented.

Other areas covered in the book include: transient utility programs, differences between DOS Plus and MS-DOS, how to check if PC software will run, the 512 disc set and much more.

At just **£9.95** this 224 page book is excellent value and is, quite simply, indispensable for Master 512 owners.

A programs disc is available at £7.95 and the book and disc can be bought together at the special price of **£14.95** (£16.95 for 3.5").

C: A Dabhand Guide

PCW said: *"I only wish this book had been available when I was learning C."* It you want to learn ANSI C then this 512 page volume is the way to do it. At **£14.95** it represents quite incredible value. Book and programs disc **£21.95**.

Best Sellers

VIEW: A Dabhand Guide £12.95
ViewSheet & ViewStore Guide £12.95
Master OS Dabhand Guide £12.95
Assembler Bundles – Two Books and two discs - machine code introduction ideal for beginners and just £9.95.

Shareware Volume 2

By popular demand a second pack of superb Master 512-tested Shareware. Another 4Mbs with a variety of programs that you have specifically asked for, including a full-function database manager with form designer, report generator and label printer.

Other programs include an excellent text editor for programmers, up-to-date file viewer, touch typing tutor and for those with their heads in the clouds a first rate astronomy package. And there's more...

Shareware Volume 2 costs just **£29.95** and if you buy Shareware volume 1 we'll throw-in an extra 6th disc of software as well!

Shareware Volume 1

Five discs full of PC software tested with the Master 512 to ensure compatibility. The collection includes software you would normally expect to pay a fortune for and includes a wordprocessor, spreadsheet, games, flowchart designer, printer utilities, and more. Price: **£29.95.**

BASIC V: A Dabhand MiniGuide

For anybody interested in BBC BASIC then this book from Beebug Editor Mike Williams is essential reading. Assuming a familiarity with BBC BASIC the various many new components of BASIC V are fully described with example programs for the Archimedes.

Even if you don't own an Archimedes the book will appeal to all BBC users who wish to remain conversant with the latest version of this powerful language. Price **£9.95**–128 pages–available end of April. Order Now!.

Mastering Interpreters and Compilers

Bruce Smith's biggest book to date and a stunner for anybody who wants to understand how their Beeb really works. In this 320-page tome, you are shown how to write both interpreters and machine code compilers - all the myths are dispelled!

As always, the text is simple and easy to follow and the programs are highly practical - in fact the compiler program would put many commercial products to shame!

SPECIAL OFFER: Beebug members - purchase this book now and we'll give you totally **FREE** the programs disc worth £7.95. Book and disc just **£14.95**. (£16.95 3.5".)

Mini Office II Guide

Bruce Smith and Robin Burton have teamed-up to write the official tutorial and reference guide to the award winning Mini Office software. A veritable mine of information it covers the everyday use of all the modules and provides information never before published. As with all Dabhand Guides the approach is practical using worked examples for you to try and liberally scattered with hints and tips.

The book covers all versions of Mini Office II in its 256 pages and is excellent value at **£9.95** or £14.95 with examples disc. (£16.95 3.5".)

New Catalogue now available FREE on Request!

DABS PRESS

5 Victoria Lane (BA), Whitefield, Manchester M25 6AL
Tel: 061-766 8423 BT Gold: 72:MAG11596 Prestel: 942876210

Prices include VAT and P&P (UK/BFPO/CI). **ACCESS/VISA** accepted by post/phone/Mailbox/ in person. Cheques and POs to address above. Dabs Press products available from all good dealers. Add £2.50 (£12 air) if outside UK. Official orders welcome.

ROUTE PLANNER EXTENDED

I was most impressed by the Route Planner (Vol.7 No.6) but thought it could be improved if a reminder of the need to refuel could be added. This might be required over long journeys such as holiday tours. Accordingly, I have added a few lines to the program to print such a reminder at approximately 200 mile intervals, and have allowed 5 minutes for this in the route timings.

```
 1010 cdist2%=0
 2345 IF cdist% MOD220>190 AND (cdist%-cd
ist2%)>30 THEN PROCpetrol ELSE needpetrol
=FALSE
 2435 IF needpetrol THEN PRINTTAB(t1)"REF
UEL NOW":ctime=ctime+5
 2800 DEF PROCpetrol
 2810 cdist2%=cdist%
 2820 needpetrol=TRUE
 2830 ENDPROC
```

There is an implication that legs of the journey do not exceed 30 miles, but if this is not the case, increase the value of 220 in line 2345 since 220-190=30.

H.Binysh

PRINTING KEYSTRIPS ON A SHINWA CP-80

The Keystrip generator (BEEBUG Vol.6 No.4 and later additions, for example Vol.6 Nos.5 & 9) appeared to be just what I want but would not print correctly on my Shinwa CP-80 printer, which is more or less Epson compatible. The main problem is the use of 1280 dots per line in double density bit image mode, and also line spacing increments of n/216 not n/256. The following amendments overcome this and may help readers with other printers which are not 100% compatible.

```
 1780 VDU27,51,20:REM spacing 20/216 inch
 2160 PROCverticalbar:PRINTMID$(legend$,1
3*keylegend%+2,12)+" ";
 2290PROCbitimage(byte2%,121):PROCbitimag
e(byte1%,2):REM 121 not 88
```

To print the keystrip title in double width characters amend line 1835 (see Vol.6 No.9) to:

```
 1835 VDU27,64,27,51,20:REM code 102 not
used on Shinwa, 20 instead of 24 for line
spacing
```

These amendments result in a strip of the right size for the keyboard.

N.L.Smith

MORE DECIMAL ARITHMETIC

The Hint given on page 65 of the December issue (Vol.7 No.7) does not work. Line 1030 should end with BCS nprt2, not BCS nprt. In any case a much more succinct version, using only 12 bytes of code, is:

```
   10 num=&70

   60 LDA num:STA &2A
   70 LDA num+1:STA &2B
   80 JMP &991F
```

The routine at &991F is the one used by Basic to print line numbers. If you still have Basic I, then this needs changing to &98F1. Locations &2A and &2B are the low and high bytes of the sixteen bit number to be printed.

T.G.Ward

Mr Ward is quite right about the error which crept into the original hint. His own alternative is undoubtedly shorter, but it is dependent on getting the correct entry point into Basic, and this is likely to be different in other versions of BBC Basic (use &A081 on the Master for instance). Note also, that this routine can only be used from within an assembler program; it cannot be CALLed from Basic.

GETTING THE REAL TIME ON A B+

The Real-Time Clock in BEEBUG Vol.7 No.8 was just what I was looking for. However, the clock vanished from the screen of my B+128 when any shadow mode was selected. The following alterations will ensure that the clock is visible in all modes.

```
 2613 .pokesc PHA:LDA sstart:STA &D6
 2616 LDA sstart+1:STA &D7:PLA:JSR &FFB3:RTS
 2640 LDY #0:JSR pokesc:JMP ewchar
 2680 .wnl LDA &140,X:JSR pokesc
 2730 JSR pokesc:INX:INY:CPY #8
 2760 ADC temp:JSR pokesc
 2850 LDY index:JSR pokesc:INC index
 2950 LDA #0:.wns LDA sblank:STA &D6
 2955 LDA sblank+1:STA &D7:LDA #0:JSR &FFB3
```

Bill Hine

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX. The normal copy date for receipt of all ads will be the 15th of each month.

WANTED: Birding Records program for BBC B Tel. (0784) 459643.

WANTED: 40/80T switchable double sided Cumana disc drive, either dual or single, must have own power supply. Tel. 041 883 9962.

BBC B with Watford DDFS iss. 7 £225. 5.25" double sided disc drive with PSU £65.3.5" disc drive £65. Green screen monitor £35. Acorn Teletext unit with ATS ROM £70. Epson MX100 wide carriage printer parallel/serial ports £125. Offers considered. Tel. (0634) 241237.

BBC Master 128 £300. 40/80T drive £100. Micro Vitec monitor £200 + printer £150. Games discs etc £50. Tel. 021-706 3285.

BBC Master 512 with Acorn Z80 2nd processor, twin discdrive, colour monitor, modem, Epson FX80 printer, software, ROMs, mags, manuals. Perfect order. Cost £2,500, want £1,100 o.n.o. Tel. (0223) 354611.

BBC B 1.2 OS with Watford DDFS and Vine Micro Replay £210. Watford 40/80T single disc drive £60. Tel. 061 962 9068.

BBC B iss. 4 + DFS, single 100K uncased discdrive, unlimited amount of software, games, utilities, etc, ROMs including (Interword, BEEBUG software). Also including Acorn and Micro User mags, User Guide and adv. User Guide. All cables supplied. £185 o.n.o. Tel. (0727) 67354.

BBC B Model B DNFS, BasicII, Micro Vitec 1431 colour monitor, 400K 40/80 switchable Teacdisc drive, Manessmann Tally MT80 printer, excellent condition, all manuals cables etc £350. Tel. (0827) 715170.

WANTED: Share Master or Share Analyser for BBC B. Tel. (0276) 22249.

Brother M-1009 dot matrix printer in original packing with user manual, friction and tractor feed, spare ribbon and BBC micro lead £65. Tel. (0252) 543244.

Interword £30, Spellmaster £30, Exmon II £10, All boxed, as new with manuals and keystrips. Tel. (0325) 463873 eves.

Business Ad
CIRCUIT DELINEATOR PROGRAM for BBC-B etc.
Ideal for projects and illustrated texts.
SAE for samples and information.
P B Helsdon
32 Burns Cresent, Chelmsford,
Essex CM2 0TS.
Tel. (0245) 251872

MiniOfficeII disc and original package, manual etc, £10 o.n.o. 20k sideways RAM board, and driver ROM, £25. Disc drive case for two half height drives, another for two full (IBM) size drives, space for PSU, £10 each. Switch joystick and cable £10, all for BBC B, all prices o.n.o. Tel. 01 253 4399 (extn 3275) or eves (0487) 814227.

BBC B iss. 7 with DNFS, Acorn Z80 2nd processor with full software and manuals, twin D/S disc drives, Philips Hi-res monitor, Watford modem and various ROMs, books and mags £425 o.n.o. complete system or offers for separate items. Tel. (0400) 61267.

DIABLO 1640 daisywheel printer, sheet and tractor feeders, acoustic cabinet and full documentation. Currently working with BBC Master. New £2000. Now only £350. Tel. (0234) 750770.

32016 software, original Cambridge workstation and co processor software at knock down prices. Reduce £135, BCPL £80, Matrix 3 £125, GKS-UK £165. Tel. (0234) 750770.

BBC Master 512 computer, DOS+ V 2.1, Mouse, GEM Write and Draw software, 11" amber monitor, CARE cartridges and SWRAM Write Protect Switch. Master reference manuals 1 and 2, DOS plus Users Guide, Master OS Book and software. Numerous other books and manuals. Master ROM, Interword ROM, Wordwise Plus ROM, plus ULTRACALC II, Novacad, Printmaster, Exmon and Speech! all in ROM. Bargain at £625. Tel. (0736) 850770 eves.

The Music System, published by Island Software. Has anyone got one that I can borrow or buy (for school usage) Tel. (0536) 67041 between 6pm-9pm.

Master128, Philips green monitor, Cumana 40/80 SS discdrive, printer lead, 50 discs in case, manuals including viewsuite and welcome discs. Other software, some games, magazines £430 o.n.o. (0280) 813934.

BBCMaster 128 with 65C102 co-processor (turbo) fitted complete with all manuals £350 o.n.o. Tubelink Advanced BASIC (BASIC V emulation) £20 o.n.o. BEEBUG Magic Modem with software and cables £100 o.n.o. All items in excellent condition. Carriage can be arranged for anywhere in the UK. Tel. (0272) 238209 9am-5.30pm or (0272) 425520 6pm onwards.

Dual disc drive, Cifer, DS/DD, 80T with integral power supply (Teac drive units) £100. Tel. Bedford 218658.

Epson FX80 printer as new £105; Sord PT351 dot matrix printer 24 pin, wide carriage £45; Sord SWP20 daisy wheel printer, wide carriage £40; New monochrome monitor complete with Nascom 1 computer with S100 bus, ROM board, RAM boards, power supply, crystal clock for RTTY reception £27;8" IBM floppy disc drive with manual £10; Trend 800ESR telex machine with keyboard, dot matrix print, LCD display etc. £37; Motorola cassette interface for 6800 development system £10; BURR Brown analogue to digital input card for 6800 system £15; Philips model 9145 stereo cassette tape deck, ok for BBC £12; TexasTI programmer calculator, decimal, hex, octal £10. Tel. (0276) 35228.

WANTED: Master128, twin 40/80 disc drives and colour monitor. Preferably midlands area. Tel. (0533) 413243.

WANTED: Nidd Valley Digimouse for master compact. Tel. (0734) 751875.

Intersheet £30, Interchart £20, Spellcheck III ROM and 80T disc £20, MasterfileII 80T£11, ATPL sideways board complete with battery backup and RAM chips £38, PMS Multifont NTQ (2 ROMS disc, guide) £24, Exmon II £17, Sleuth £12, Norwich CPROM III £12, Design (tape) £5, all boxed and complete. Red boxes project manual £5.Tel. (0684) 572295.

WANTED: M512K module, with or without Co-pro adaptor. For BBC B. Tel. (0526) 21539.

Plain fanfold paper 11" x 14.5" (across) 7 boxes x 2000 each. £7 per box or £45 the lot. Tel. 01-864 1614.

Business Ad

SPROG SQUASHER

Reduce size and increase speed of Wordwise-Plus segment programs. On 5.25 inch disc/cassette for BBC Micro. Only £5 (incl p&p). Cheque/PO payable to C G Robbins.

Send for details or order to

Round Robin.
29 Sandpit Lane, St. Albans,
Herts, AL1 4EW.
Tel. (0727) 30264

512 users,

Solve the compatibility problems of your MASTER 512 or BBC with co-pro adaptor using DOS+ Problem Solver

It corrects hardware incompatibilities (such as programmable interrupts' speed, low level keyboard scanning, etc.) and operating system's bugs, enabling the 512 board to run most IBM-PC programs that otherwise wouldn't run.

Lots of programs like Cat, Jet, Digger, ARTWORX's Strip Poker, ELECTRONIC ART's Golf, Driller, Dark Side, Impact, Charlie Chaplin, Test Drive, Infiltrator, StarQuake, Bushido, Tennis, all versions of MicroSoft's Flight Simulator, Frogger, Osbit, 688 Atack Sub, Defender of the Crown, Quadralian, Yes Chancellor, Anciant Art of War, Adventure Writter, Dream House, AFT, Droege, Dream, Delux Paint 2, Fontasy 2.08, News, PC Tutor, Lotus 123 2.0, Turbo Calc, Mandelbrot Generator, Prospero Pascal, Turbo Pascal 4.0, Turbo C 1.5, Turbo Prolog, Prolog2, PC File+, Galaxy, Trendtex/2, Homebase 2.15, Mindreader, DBASE III plus, Pipedream, etc. will run like in an ordinary IBM PC.

The program provides the INS, DEL, PG-UP, PG-DOWN, HOME, END and SC-LOCK keys to users who don't have the numeric keypad.
It will also allow you to switch between colour and B&W modes, change the computer's speed and save the hi-res picture on the screen, during the execution of any program.

DOS+ Problem Solver works with all 80186 co-processor boards (with 512K or 1M bytes RAM) using DOS+ 1.2-XIOS 1.00, DOS+ 1.2-XIOS 1.01 or DOS+ 2.1-XIOS 1.03.
If you need any further informations about DOS+ Problem Solver, please contact Shibumi Soft .

PRICE: £24.95 inc. program, user manual & one year free user suport.
Price includes VAT. Please add £3 for postage and packing. Faulty disks will be replaced.
Cheques should be payable to Shibumi Soft .

Shibumi Soft: R. Prof. Camara Sinval, 138
4100 PORTO
PORTUGAL

WANTED: for a BBC Micro, ICs Type 5C094 (ULA), 2C199(ULA), 6845 (Video Gen), and SAA 50 50 (CharGen) or equivalents. Tel. (0745) 825036.

KAGA 12" amber monitor £30, Decca 80 16" colour TV modified to RGB monitor, as supplied by Display Electronics £25. Tel. (0582) 581051.

WANTED: Computer Concepts MEGA 3 ROM and Spellmaster ROM. Manuals etc must be included. Tel. (0253)67987 after 7.30pm.

BBC B working, but with some superficial damage. Offers over £100 - to be donated to school for handicapped children. Tel. (025 485) 3807.

BBC computer with DFS, joystick and light pen £225, Viglen PC case with Viglen ROM cartridge system £25, D/S 40T D/D £60, D/S 40TD/D (uncased) £40, Colour monitor £130, Teletext adaptor with latest ROM £50, ATPL ROMboard including 16k RAM, printer buffer, utilities and ROM £25, Watford shadow RAM £35, Spellmaster £35, Interword £30, Interbase £35 (latest), 100discs (many containing software), EPROMS 27128's 16k, 21 volt £3.5 each and many other goodies. Tel. (0332) 556381.

512 board with DOS+, mouse and GEM to "PC" the Master £75 o.n.o. also Exmon II ROM £12, Advanced Disc Investigator ROM £12 and Elite pack £6. Tel. (0222) 705304.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES		BEEBUG & RISC USER
£ 7.50	6 months (5 issues) UK only	
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£23.00
£20.00	Rest of Europe & Eire	£33.00
£25.00	Middle East	£40.00
£27.00	Americas & Africa	£44.00
£29.00	Elsewhere	£48.00

BACK ISSUE PRICES (per Issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

POST AND PACKING
Please add the cost of p&p as shown opposite.

BEEBUG
Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Technical Assistant: Glynn Clements
Advertising: Sarah Shrive
Production Assistant: Sheila Stoneman
Membership secretary: Mandy Mileham
Managing Editor: Sheridan Williams

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1989
ISSN - 0263 - 7561
Printed by Newnorth-Burt Ltd (0234) 41111

Magazine Disc/Cassette

APRIL 1989 DISC/CASSETTE CONTENTS

INDEXING DFS FORMAT DISCS - use this program to compile a handy index to all your programs.

MODE 7 HISTOGRAMS - display up to four graphs together on screen with this versatile and flexible application.

FIVE NEW SCREEN MODES - implement a new star command to give yourself a choice of modes 8 to 12.

UNDERSTANDING AC CIRCUITS - the latest application in our educational series dealing with the concepts of RC and LR circuits.

FIRST COURSE - three programs demonstrating the use of teletext control codes.

A SCREEN PRINTER DRIVER FOR VIEW - a short utility for users of View to allow text highlights (bold and underline) to be seen on screen.

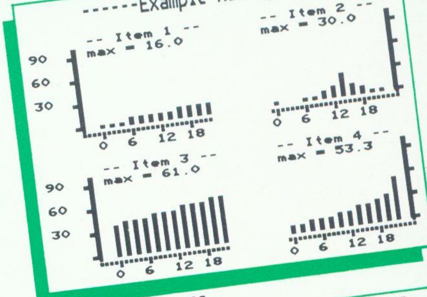
PAGE COMPOSITION FOR THE BBC MICRO (Part 2)- five programs in all comprising a screen dump, choice of two printer drivers and two font generators to conclude our presentation on page layouts.

MULTI-PRECISION DECIMAL ARITHMETIC (Part 2) - the complete suite of decimal arithmetic routines, and a separate Basic procedure for ease of use.

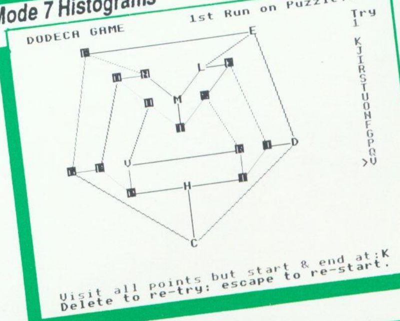
THE DODECA GAME - a test of memory and mental agility in this unusual game.

MAGSCAN - bibliography for this issue (Vol.7 No.10).

VIDEO CASSETTE CATALOGUER - an additional item in the form of an improved version of the program first published in Vol.6 Nos.9 &10.

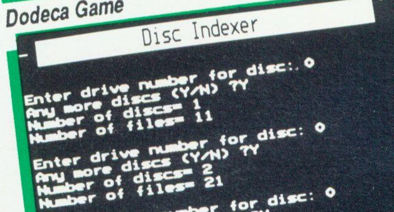


Mode 7 Histograms



```
Visit all points but start & end at:K
Delete to re-try: escape to re-start.
```

Dodeca Game



```
Disc Indexer

Enter drive number for disc: 0
Any more discs (Y/N) ?Y
Number of discs= 1
Number of files= 11

Enter drive number for disc: 0
Any more discs (Y/N) ?Y
Number of discs= 2
Number of files= 21

Enter drive number for disc: 0
Any more discs (Y/N) ?Y
Number of discs= 3
Number of files= 32
```

Indexing DFS Format Discs

All this for £3. 50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY 5" Disc	UK ONLY 3.5" Disc	UK ONLY Cassette	OVERSEAS 5" Disc	OVERSEAS 3.5" Disc	OVERSEAS Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:
BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

BEEBUG Discs - The Ultimate in Quality & Reliability

80 Track Double Sided Quad Density

	Members Price	Order Code
10	£9.90	0660
25	£24.90	0664
50	£39.90	0668

Free disc boxes

25/50 discs with free lockable storage box

Prices shown are members prices and include VAT.

BEEBUG discs are manufactured to the highest specifications and are fully guaranteed.

POSTAGE

Boxes of ten discs £2 Post code C

Boxes of 25 or more £4 Post code E

40 Track Single Sided Double Density

	Members Price	Order Code
10	£8.90	0657
25	£21.85	0661
50	£35.60	0665

3.5" Double Sided Double Density

	Members Price	Order Code
10	£15.00	0675
40	£56.00	0676

Our Guarantee

We confidently offer a lifetime data guarantee and will replace any disc with which you encounter problems. We have found that the standard of quality control at the factory makes this necessity very rare.

Please send me ______ (qty) ______ (stock code) at £______ (unit price)

UK post 10 £2, 25/40/50 £4. See retail catalogue for Overseas postage rates

I enclose a cheque for £______ /Please debit my Access/Visa card £________

Expiry date:

Name ________________ Memb No. ________________

Address ________________________________

____

BEEBUG

Dolphin Place,
Holywell Hill,
St. Albans,
Herts.
AL1 1EX

☎ (0727) 40303